W3School jQuery UI 教程

wizardforcel

Published with GitBook



目錄

W3School jQuery UI教程

来源：

- jQuery UI教程
- jQuery Easy UI教程

整理：飞龙

jQuery UI 基础

jQuery UI 简介

jQuery UI 是一个建立在 jQuery JavaScript 库上的小部件和交互库，您可以使用它创建高度交互的 Web 应用程序。本教程将向您讲解 jQuery UI 是如何工作的。

jQuery UI 特性

简单易用

继承 jQuery 简易使用特性，提供高度抽象接口，短期改善网站易用性。

开源免费

采用 MIT & GPL 双协议授权，轻松满足自由产品至企业产品各种授权需求。

广泛兼容

兼容各主流桌面浏览器。包括IE 6+、Firefox 2+、Safari 3+、Opera 9+、Chrome 1+。

轻便快捷

组件间相对独立，可按需加载，避免浪费带宽拖慢网页打开速度。

标准先进

支持 WAI-ARIA，通过标准 XHTML 代码提供渐进增强，保证低端环境可访问性。

美观多变

提供近 20 种预设主题，并可自定义多达 60 项可配置样式规则，提供 24 种背景纹理选择。

开放公开

从结构规划到代码编写，全程开放，文档、代码、讨论，人人均可参与。

强力支持

Google 为发布代码提供 CDN 内容分发网络支持。

完整汉化

开发包内置包含中文在内的 40 多种语言包。

缺点、不足

- 1.代码不够健壮：缺乏全面的测试用例，部分组件 Bugs 较多，不能达到企业级产品开发要求。
- 2.构架规划不足：组件间 API 缺乏协调，缺乏配合使用帮助。
- 3.控件较少：相对于 Dojo、YUI、Ext JS 等成熟产品，可用控件较少，无法满足复杂界面功能要求。

jQuery UI 下载

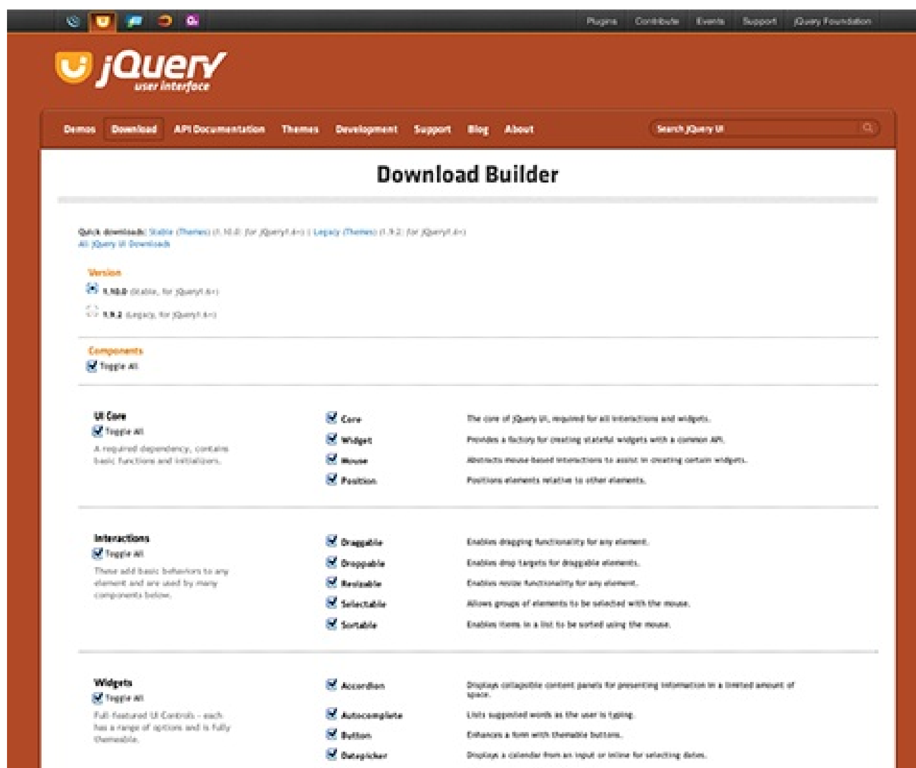
一旦您对 jQuery UI 有了基本了解，您就可以亲自尝试一下。请从 jQuery UI 网站上的 Download Builder（下载生成器） 页面下载 jQuery UI 的副本。

创建自定义 jQuery UI 下载

jQuery UI 的下载生成器（Download Builder）允许您选择您需要下载的组件，为项目获取一个自定义的库版本。创建自定义 jQuery UI 下载需要以下三个步骤：

步骤 1：选择您需要的组件

下载生成器（Download Builder）页面的第一栏列出了 jQuery UI 所有的 JavaScript 组件分类：核心（UI Core）、交互部件（Interactions）、小部件（Widgets）和效果库（Effects）。jQuery UI 中的一些组件依赖于其他组件，当选中这些组件时，它所依赖的其他组件也都会自动被选中。您所选的组件将会合并到一个 jQuery UI JavaScript 文件。



步骤 2：选择一个主题或者自定义一个主题

在下载生成器（Download Builder）页面，您将看到一个文本框，列出了一系列为 jQuery UI 小部件预先设计的主题。您可以从这些提供的主题中选择一个，也可以使用 ThemeRoller 自定义一个主题（详见后面章节的讲解）。

高级主题设置： 下载生成器（*Download Builder*）的主题部分也为主题提供了一些高级配置设置。如果您打算在一个页面上使用多个主题，这些字段会派上用场。如果您打算在一个页面上只使用一个主题，那么您完全可以跳过这些设置。

步骤 3：选择 jQuery UI 的版本

在下载生成器（Download Builder）中，最后一步是选择一个版本号。这个步骤很重要，因为 jQuery UI 的版本是配合特定的 jQuery 版本设计的。目前的版本有：

- jQuery UI 1.10.2 – 要求 jQuery 1.6 及以上版本。
- jQuery UI 1.9.2 – 要求 jQuery 1.6 及以上版本。

点击 Download 按钮进行下载！

点击 Download 按钮，完成下载。您将得到一个包含您所选组件的自定义 zip 文件。

jQuery UI 使用

一旦您下载了 jQuery UI，您将得到一个 zip 压缩包，包含下列文件：

- `/css/`
- `/development-bundle/`
- `/js/`
- `index.html`

在网页上使用 jQuery UI

在文本编辑器中打开 `index.html`，您将看到引用了一些外部文件：主题、jQuery 和 jQuery UI。通常情况下，您需要在页面中引用这三个文件，以便使用 jQuery UI 的窗体小部件和交互部件：

```
<link rel="stylesheet" href="css/themename/jquery-ui.custom.css" />
<script src="js/jquery.min.js"></script>
<script src="js/jquery-ui.custom.min.js"></script>
```

一旦您引用了这些必要的文件，您就能向您的页面添加一些 jQuery 小部件。比如，要制作一个日期选择器（datepicker）小部件，您需要向页面添加一个文本输入框，然后再调用 `.datepicker()`，如下所示：

HTML:

```
<input type="text" name="date" id="date" />
```

JavaScript:

```
$( "#date" ).datepicker();
```



如需查看 jQuery UI 小部件和交互部件的实例演示，请访问 jQuery UI 实例。

jQuery UI 定制

jQuery UI 提供了多种定制方式。您已经看到下载生成器（Download Builder）如何定制一个值包含您所需选项的自定义版本，这里还提供了其他定制方式。

jQuery UI 基础：使用选项

jQuery UI 中的每个插件都有一个默认配置，默认配置值一般是根据最基本最常见的使用情况设置的。如果您想要让某个插件设置成非默认值，您可以使用 "options" 重写它的默认设置。选项是一组属性，作为参数传递给 jQuery UI 小部件。例如，滑块（slider）小部件具有 orientation 选项，该选项允许您指定滑块是水平的还是垂直的。为了设置滑块的该选项，您只需将它作为一个参数传递，如下所示：

```
$( "#mySliderDiv" ).slider({
    orientation: "vertical"
});
```

您可以传递更多不同的选项，每个选项之间用逗号分隔：

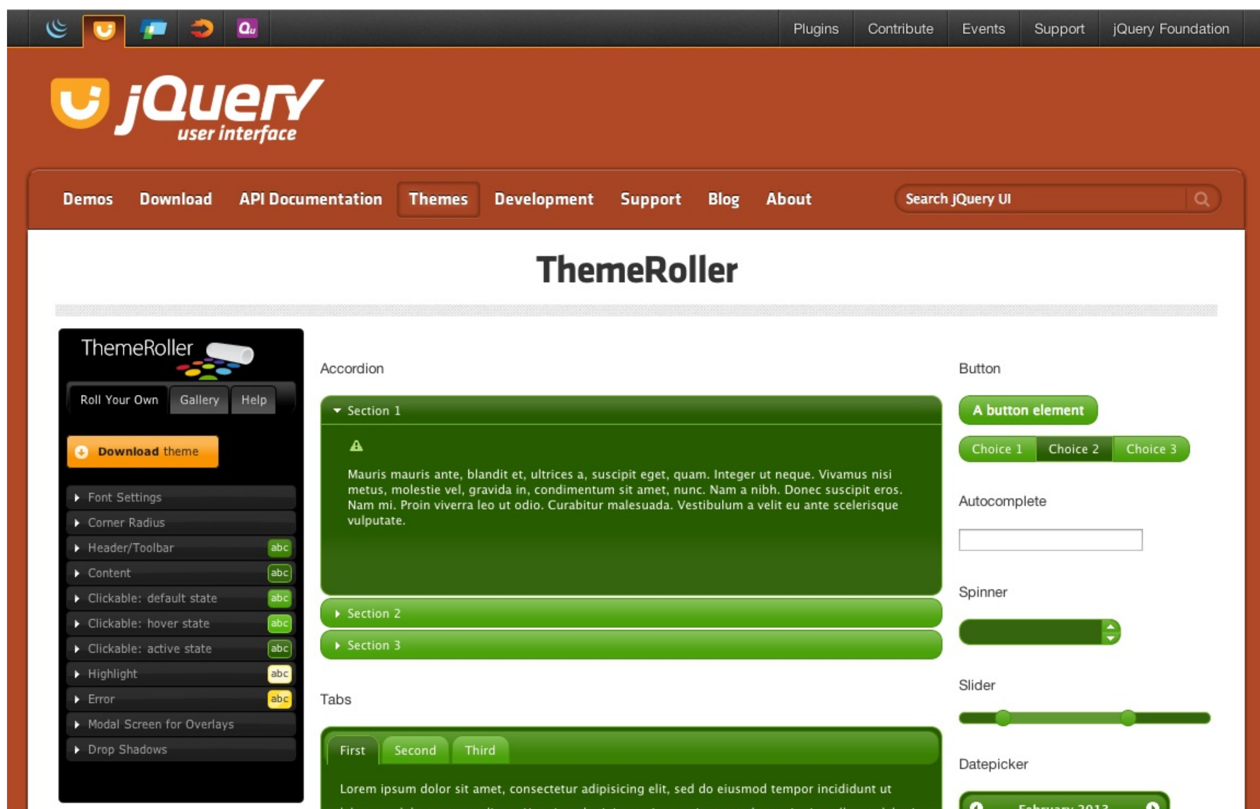
```
$( "#mySliderDiv" ).slider({
    orientation: "vertical",
    min: 0,
    max: 150,
    value: 50
});
```

请记得选项需放在大括号 `{ }` 内。上面的实例只是一个简单的讲解，如需获取整套 jQuery UI 小部件的详细信息，请查看 jQuery UI 实例。

视觉定制：设计一个 jQuery UI 主题

如果您想要设计自己的主题，jQuery UI 提供了一个非常完美的用于主题定制的应用程序，这就是 ThemeRoller。具体定制请访问 jQuery UI ThemeRoller。

ThemeRoller 为所有使用 jQuery UI 小部件设计的元素提供了一个自定义接口。当您调整左栏中的"levers"，右栏中的小部件将根据您的设计进行显示。ThemeRoller 的 Gallery 选项卡提供了一些与设计主题，与下载生成器（Download Builder）页面提供的一样。您可以基于这些主题做调整，或者直接下载。



下载主题

当您点击 ThemeRoller 页面中的 "Download theme" 按钮，将跳转到下载生成器（Download Builder）页面，您的自定义主题会在主体下拉菜单中自动选中。您可以进一步配置下载包。一旦下载完成，您将看到 `example.html` 页面使用了您自定义的主题。

提示： 如果您需要编辑您的主题，只需打开 *CSS* 文件，找到第 *43* 行，*"To view and modify this theme, visit ..."*，该 *url* 即为在 *ThemeRoller* 中打开主题进行编辑的链接。

jQuery UI 工作原理

jQuery UI 包含了许多维持状态的小部件（Widget），因此，它与典型的 jQuery 插件使用模式略有不同。其安装方式与大部分 jQuery 插件的安装方式类似，jQuery UI 的小部件是基于 部件库（Widget Factory） 创建的，小部件库提供了通用的 API。所以，只要您学会使用其中一个，您就知道如何使用其他的小部件（Widget）。本教程将通过 进度条（progressbar） 小部件代码实例介绍常见的功能。

安装

为了跟踪部件的状态，我们首先介绍一下小部件的全生命周期。当小部件安装时，生命周期开始。我们只需要在一个或多个元素上调用插件，即安装了小部件。

```
$( "#elem" ).progressbar();
```

这将会初始化 jQuery 对象中的每个元素，在本例中，元素 id 为 "elem"。因为我们调用无参数的 `.progressbar()` 方法，小部件则会按照它的默认选项进行初始化。我们可以在安装时传递一组选项，这样既可重写默认选项。

```
$( "#elem" ).progressbar({ value: 20 });
```

安装时传递的选项数目多少可根据我们的需要而定。任何我们未传递的选项则都使用它们的默认值。

选项是小部件状态的组成部分，所以我们也可以在安装后再进行设置选项。我们将在后续的 `option` 方法中介绍这部分内容。

方法

既然小部件已经初始化，我们就可以查询它的状态，或者在小部件上执行动作。所有初始化后的动作都以方法调用的形式进行。为了在小部件上调用一个方法，我们可以向 jQuery 插件传递方法的名称。例如，为了在进度条（progressbar）小部件上调用 `value` 方法，我们应该使用：

```
$( "#elem" ).progressbar( "value" );
```

如果方法接受参数，我们可以在方法名后传递参数。例如，为了传递参数 `40` 给 `value` 方法，我们可以使用：

```
$( "#elem" ).progressbar( "value", 40 );
```

就像 jQuery 中的其他方法一样，大部分的小部件方法为链接返回 jQuery 对象。

```
$( "#elem" )
    .progressbar( "value", 90 )
    .addClass( "almost-done" );
```

公共的方法

每个小部件都有它自己的一套基于小部件所提供功能的方法。然而，有一些方法是所有小部件都共同具有的。

option

正如我们前面所提到的，我们可以在初始化之后通过 `option` 方法改变选项。例如，我们可以通过调用 `option` 方法改变 progressbar（进度条）的 value 为 30。

```
$( "#elem" ).progressbar( "option", "value", 30 );
```

请注意，这与之前我们调用 `value` 方法的实例有所不同。在本实例中，我们调用 `option` 方法，改变 value 选项为 30。

我们也可以为某个选项获取当前的值。

```
$( "#elem" ).progressbar( "option", "value" );
```

另外，我们可以通过给 `option` 方法传递一个对象，一次更新多个选项。

```
$( "#elem" ).progressbar( "option", {
    value: 100,
    disabled: true
});
```

您也许注意到 `option` 方法有着与 jQuery 代码中取值器和设置器相同的标志，就像 `.css()` 和 `.attr()` 。唯一的不同就是您必须传递字符串 "option" 作为第一个参数。

disable

`disable` 方法禁用小部件。在进度条（progressbar）实例中，这会改变样式让进度条显示为禁用状态。

```
$( "#elem" ).progressbar( "disable" );
```

调用 `disable` 方法等同于设置 `disabled` 选项为 `true` 。

enable

`enable` 方法是 `disable` 方法的对立面。

```
$( "#elem" ).progressbar( "enable" );
```

调用 `enable` 方法等同于设置 `disabled` 选项为 `false` 。

destroy

如果您不再需要小部件，那么可以销毁它，返回到最初的标记。这意味着小部件生命周期的终止。

```
$( "#elem" ).progressbar( "destroy" );
```

一旦您销毁了一个小部件，您就不能在该部件上调用任何方法，除非您再次初始化这个小部件。如果您要移除元素，可以直接通过 `.remove()` ，也可以通过 `.html()` 或 `.empty()` 修改祖先，小部件会自动销毁。

widget

一些小部件生成包装器元素，或与原始元素断开连接的元素。在下面的实例中， `widget` 将返回生成的元素。在进度条（progressbar）实例中，没有生成的包装器， `widget` 方法返回原始的元素。

```
$( "#elem" ).progressbar( "widget" );
```

事件

所有的小部件都有跟他们各种行为相关的事件，用于在状态改变时通知您。对于大多数的小部件，当事件被触发时，名称以小部件名称为前缀。例如，我们可以绑定进度条（）的 change 事件，一旦值发生变化时就触发。

```
$( "#elem" ).bind( "progressbarchange", function() {
    alert( "The value has changed!" );
});
```

每个事件都有一个相对应的回调，作为选项进行呈现。我们可以使用进度条（progressbar）的 `change` 回调，这等同于绑定 `progressbarchange` 事件。

```
$( "#elem" ).progressbar({
    change: function() {
        alert( "The value has changed!" );
    }
});
```

公共的事件

大多数事件是针对特定的小部件，所有的小部件都有一个公共的 `create` 事件。该事件在小部件被创建时即被触发。

jQuery UI 主题

jQuery UI 主题

所有的 jQuery UI 插件都允许开发人员无缝集成 UI 小部件到他们网站或应用程序的外观和感观。每个插件通过 CSS 定义样式，且包含了两层样式信息：标准的 jQuery UI CSS 框架 样式和具体的插件样式。

jQuery UI CSS 框架提供了语义表示的类，用来表明小部件内元素的角色，比如标题、内容或可点击区域。这些在所有的小部件中都是一致的，一个可点击的 tab（标签页）、accordion（折叠面板）或 button（按钮）都有相同的 `ui-state-default` class，用来表明它们是可点击的。当用户鼠标悬浮在这些元素上面时，这个 class 就变成 `ui-state-hover` ，当选中这些元素时则变成 `ui-state-active` 。这些 class 的一致性使得所有部件中具有相似角色或交互状态的元素在外观表现上一致。

CSS 框架样式封装在一个单独的文件中，名为 `ui.theme.css` 。这个文件时通过 ThemeRoller 应用程序来修改的。框架样式只包含影响外观和感观的属性，只要是颜色、背景图像、图标等。所以这些是 "安全的" 样式，不会影响到插件的功能。这种分隔意味着开发人员可以通过在 `theme.css` 文件中修改颜色和图像来创建一个自定义的外观和感观。由于未来的插件或者 bug 修复将是可用的，这些不通过修改即可与主题一起使用。

由于框架样式只覆盖了外观和感观，所以还需要包含具体的插件样式表，这些样式表包括了所有额外的让小部件具有功能性的结构样式规则，比如尺寸、内边距、外边距、定位、浮动。每个插件的样式表位于 `themes/base` 文件夹内，且配合插件进行命名，比如 "jquery.ui.accordion.css"。这些样式必须认真编辑，因为它们与脚本一起提供了框架样式的覆盖。

我们鼓励所有的开发人员创建 jQuery 插件，jQuery UI CSS 框架使得最终用户更容易定制主题和使用插件。

主题化

下面列出了三种主题化 jQuery UI 插件的一般方法：

- 下载 **ThemeRoller** 主题：最早的创建主题的方式是使用 ThemeRoller 来生成和下载一个主题。这个应用程序将创建一个新的 `ui.theme.css` 文件和一个包含了所有必需的背景图像及图标精灵的 `images` 文件夹。这个方法是最早的创建和维护主题的方式，但是它对 ThemeRoller 中提供的选项限制了自定义。
- 修改 **CSS** 文件：为了对外观和感观做进一步的控制，您可以选择从默认主题（Smoothness）开始，或者从一个由 ThemeRoller 生成的主题开始，然后调整 `ui.theme.css` 文件，或者任意一个独立插件的样式表。例如，您可以很容易地调整所有按钮的角半径为不同于其他 UI 组件的值，或者使用自定义设置为图标精灵改变路径。通过一点点的样式范围，您甚至可以在一个 UI 中同时使用多个主题。为了易于维护，建议只更改 `ui.theme.css` 文件和图像。
- 重新编写自定义的 **CSS**：为了最大程度地控制外观和感观，可以重新开始编写

每个插件的 CSS，而不使用框架类或者特定的插件样式表。如果想要的外观和感观不能通过修改 CSS 来实现或者使用高度自定义的标记，那么就可以采用这个方法。这个方法要求在 CSS 方面有深厚的专业知识，且要求手动更新未来的插件。

使用 **ThemeRoller**、**jQuery UI CSS** 框架，以及设计自定义主题

- jQuery UI ThemeRoller
- jQuery UI CSS 框架 API
- 设计主题

jQuery UI ThemeRoller

ThemeRoller 简介

ThemeRoller 是一个 Web 应用程序，为 jQuery UI 设计和下载自定义主题提供了直观的界面。您可以访问 jQuery UI ThemeRoller 进行主题定制。

jQuery UI ThemeRoller 是由波士顿的 Filament Group, Inc 设计和开发的。



ThemeRoller 界面

ThemeRoller 的界面分为不同面板，各面板分别是全局字体和圆角半径设置、小部件容器样式、可点击元素的互动状态，及覆盖和阴影的各种样式。这些面板允许配置各种 CSS 属性，比如字体尺寸、颜色、粗细、背景颜色和纹理、边框颜色、文本颜色、图标颜色、圆角半径，等等。

主题馆（Gallery）：预先设计主题

ThemeRoller 主题可以通过永久链接 URL 进行查看，它包含一些预先设计的主题可供选择。主题馆（Gallery）可以通过位于应用程序界面顶端的标签栏进行访问。从主题馆（Gallery）中，您可以预览和下载主题，甚至可以选择一个主题，然后切换到 "Roll Your Own" 标签页进行调整。

下载主题

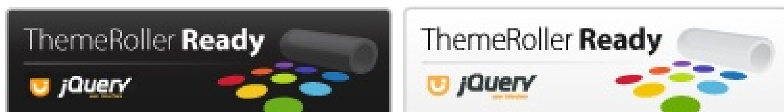
当您设计完主题后，您可以下载主题以便在项目中进行使用。ThemeRoller 在顶部有一个 "Download theme（下载主题）" 按钮，可以生成一个 zip 压缩包，包含了所有的主题相关文件。下载文件中的图像是按照您的规格进行生成的，并保存为高质量的 PNG 文件。

您的主题将包含图像和 CSS，组成了 jQuery UI CSS 框架的自定义版本，包含了所有插件的图像和 CSS。

在项目中按照下载的主题

一旦下载了主题，解压缩后，您将看到一个名为 `themes` 的文件夹。该文件夹包括了主题所需的 CSS 和图像。复制主题文件夹到您的项目中，并在页面中链接 `themes/all.css` 文件。

创建自定义的 "ThemeRoller-Ready" 组件



ThemeRoller 生成了一个 jQuery UI CSS 框架的自定义版本，用于开发您自己的 ThemeRoller-ready jQuery 组件。通过这个框架生成的类被设计来提供通用的用户界面的设计情况，包括状态、图标以及各种辅助类。

如需了解更多 jQuery UI CSS 框架的开发信息，请查看 主题化 API 文档。

ThemeRoller 链接

- 教程：开发您自己的 jQuery "ThemeRoller-Ready" 组件，Filament Group
- ThemeRoller 简介：设计 & 下载 jQuery UI 自定义主题，Filament Group
- 视频：设计 & 下载 jQuery UI 自定义主题，Filament Group

jQuery UI CSS 框架 API

jQuery UI CSS 框架

jQuery UI 包含了一个强大的 CSS 框架，为了创建自定义 jQuery 小部件而设计的。框架包含了通用的用户界面所需的类，且可使用 jQuery UI ThemeRoller 进行维护。通过使用 jQuery UI CSS 框架创建您自己的 UI 组件。您需采用共享标记公约，以便在插件社区的代码集成。

框架类

下面的 CSS 类根据样式是否是固定的结构化的，或者是否是可主题化的（颜色、字体、背景等），分别定义在 `ui.core.css` 和 `ui.theme.css` 两个文件中。这些类被设计来用于用户界面元素，以便获得整个应用程序的视觉一致性，可通过 jQuery UI ThemeRoller 对组件进行主题化。

布局助手

- `.ui-helper-hidden` ：对元素应用 `display: none` 。
- `.ui-helper-hidden-accessible` ：对元素应用访问隐藏（通过页面绝对定位）。
- `.ui-helper-reset` ：UI 元素的基本样式重置。重置的元素比如： `padding` 、 `margin` 、 `text-decoration` 、list-style，等等。
- `.ui-helper-clearfix` ：对父元素应用浮动包装属性。
- `.ui-helper-zfix` ：对 `<iframe>` 元素应用 iframe "fix" CSS。

小部件容器

- `.ui-widget` ：对所有小部件的外部容器应用的 Class。对小部件应用字体和字体尺寸，同时也对自表单元素应用相同的字体和 1em 的字体尺寸，以应对 Windows 浏览器中的继承问题。
- `.ui-widget-header` ：对标题容器应用的 Class。对元素及其子元素的文本、链接、图标应用标题容器样式。
- `.ui-widget-content` ：对内容容器应用的 Class。对元素及其子元素的文本、链接、图标应用内容容器样式。（可应用到标题的父元素或者同级元素）

交互状态

- `.ui-state-default` ：对可点击按钮元素应用的 Class。对元素及其子元素的文本、链接、图标应用 "clickable default" 容器样式。
- `.ui-state-hover` ：当鼠标悬浮在可点击按钮元素上时应用的 Class。对元素及其子元素的文本、链接、图标应用 "clickable hover" 容器样式。
- `.ui-state-focus` ：当键盘聚焦在可点击按钮元素上时应用的 Class。对元

素及其子元素的文本、链接、图标应用 "clickable hover" 容器样式。
- `.ui-state-active` ：当鼠标点击可点击按钮元素上时应用的 Class。对元素及其子元素的文本、链接、图标应用 "clickable active" 容器样式。

交互提示 Cues

- `.ui-state-highlight` ：对高亮或者选中元素应用的 Class。对元素及其子元素的文本、链接、图标应用 "highlight" 容器样式。
- `.ui-state-error` ：对错误消息容器元素应用的 Class。对元素及其子元素的文本、链接、图标应用 "error" 容器样式。
- `.ui-state-error-text` ：对只有无背景的错误文本颜色应用的 Class。可用于表单标签，也可以对子图标应用错误图标颜色。
- `.ui-state-disabled` ：对禁用的 UI 元素应用一个暗淡的不透明度。意味着对一个已经定义样式的元素添加额外的样式。
- `.ui-priority-primary` ：对第一优先权的按钮应用的 Class。应用粗体文本。
- `.ui-priority-secondary` ：对第二优先权的按钮应用的 Class。应用正常粗细的文本，对元素应用轻微的透明度。

图标

状态和图像

- `.ui-icon` ：对图标元素应用的基本 Class。设置尺寸为 16px 方块，隐藏内部文本，对 "content" 状态的精灵图像设置背景图像。注意： *`.ui-icon` class* 将根据它的父容器得到一个不同的精灵背景图像。例如， *`ui-state-default`* 容器内的 *`ui-icon`* 元素将根据 *`ui-state-default`* 的图标颜色进行着色。

图标类型

在声明 `.ui-icon` class 之后，接着您可以声明一个秒速图标类型的 class。通常情况下，图标 class 遵循语法 `.ui-icon-{icon type}-{icon sub description}-{direction}` 。

例如，一个指向右侧的三角形图标，如下所示： `.ui-icon-triangle-1-e`

jQuery UI 的 ThemeRoller 在它的预览一栏中提供了全套的 CSS 框架图标。将鼠标悬浮在图标上可查看 class 名称。

其他视觉效果

圆角半径助手

- `.ui-corner-tl` ：对元素的左上角应用圆角半径。

- `.ui-corner-tr` ：对元素的右上角应用圆角半径。
- `.ui-corner-bl` ：对元素的左下角应用圆角半径。
- `.ui-corner-br` ：对元素的右下角应用圆角半径。
- `.ui-corner-top` ：对元素上边的左右角应用圆角半径。
- `.ui-corner-bottom` ：对元素下边的左右角应用圆角半径。
- `.ui-corner-right` ：对元素右边的上下角应用圆角半径。
- `.ui-corner-left` ：对元素左边的上下角应用圆角半径。
- `.ui-corner-all` ：对元素的所有四个角应用圆角半径。

覆盖 & 阴影

- `.ui-widget-overlay` ：对覆盖屏幕应用 100% 宽度和高度，同时设置背景颜色/纹理和屏幕不透明度。
- `.ui-widget-shadow` ：对覆盖应用的 Class，设置了不透明度、上偏移/左偏移，以及阴影的 "厚度"。厚度是通过对阴影所有边设置内边距（padding）进行应用的。偏移是通过设置上外边距（margin）和左外边距（margin）进行应用的（可以是正数，也可以是负数）。

jQuery UI 设计主题

文件结构

主题是以特定的方式来增加他们的易用性。通常，文件目录结构如下所示：

- `themename/` – 您的主题必须完全包含在一个单独的以主题名称命名的文件夹内。
- `themename/themename.css` – 这是基本的 CSS 文件。无论使用了哪个插件，该文件都必须在每个使用主题的页面中引用。该文件应该是轻量级的，只包括要点。
- `themename/themename.pluginname.css` – 您支持的每个插件都需要一个 CSS 文件。插件的名称应直接包含在文件名中。例如，如果您为 tabs（标签页）插件进行主题化，则有： `themename.tabs.js` 。
- `themename/img.png` – 您的主题可以包含图像。它们可以根据您的喜好进行命名，这里没有特定的命名惯例。

如需了解主题文件结构是如何完成的实例，请访问 jQuery UI 基本主题。

定义样式

为主题编写样式是非常简单的，这是因为主题的灵活性。

所有的主题都应该有一个基本的 CSS class。这个主要的 class 允许用户启用禁用主题。您的根 class 的格式应该是 `.ui-themename` 。且它在 HTML 文件中的用法如下所示：

```
<html>
<head>
    <title>My Site</title>
    <link rel="stylesheet" href="themename/themename.css" />
    <link rel="stylesheet" href="othertheme/othertheme.css" />
    <link rel="stylesheet" href="othertheme/othertheme.dialog.css"
</head>
<body class="ui-themename">
    <div class="ui-othertheme">
        <div class="ui-dialog">This is a modal dialog.</div>
    </div>
</body>
</html>
```

在上面的实例中，发生了一些重要的事情：

- 我们同时向文档中加载两个主题。
- 整个页面机器所有内容，是根据 themename 的样式进行主题化的。

- 然而，带有 ui-othertheme class 的 <div> 及其中的每个元素（包括模态对话框）都是根据 othertheme 的样式进行主题化的。

如果我们打开 `themename.css` 文件进行查看，我们可以看到如下代码：

```
body.ui-themename { background:#111; color:snow; }
.ui-themename a, a.ui-themename { color:#68D; outline:none; }
.ui-themename a:visited, a.ui-themename:visited { color:#D66; }
.ui-themename a:hover, a.ui-themename:hover { color:#FFF; }
```

请注意，`themename.css` 文件只包括全局通用的样式信息，特定插件的样式信息不在这里进行定义。这里的样式对所有主题都是适用的。不用担心一个主题会占据多个文件 - 这些会在创建和下载的过程被简化。

jQuery UI 部件库

jQuery UI 部件库（Widget Factory）

jQuery UI 部件库（Widget Factory）是一个可扩展的基础，所有的 jQuery UI 小部件都是在上面进行创建的。使用部件库（Widget Factory）来创建插件，提供了方便的状态管理，同时也为一些常见的任务提供了便捷，比如暴露插件方法，实例化后改变选项等。

- 通过部件库（Widget Factory）扩展小部件（Widget）
- 小部件（Widget）方法调用
- 为什么使用部件库（Widget Factory）
- 如何使用部件库（Widget Factory）

jQuery UI 通过部件库（Widget Factory）扩展小部件

jQuery UI 的部件库（Widget Factory）使得创建小部件变得更加容易，这些小部件扩展了已有小部件的功能。这样子您就能在已有的基础上创建出功能强大的小部件，也可以在已有的小部件功能上做细微的调整。

注意：在学习本章节之前，需要明白什么是部件库（Widget Factory），及它是怎么工作的。如果您对这些知识还不熟悉，那么请先查看如何使用部件库（Widget Factory）章节。

创建小部件扩展

通过部件库（Widget Factory）创建小部件是通过向 `$.widget()` 传递小部件名称和一个原型对象来完成的。下面的实例是在 "custom" 命名空间中创建一个 "superDialog" 小部件。

```
$.widget( "custom.superDialog", {} );
```

为了支持扩展， `$.widget()` 可选性地接受作为父部件使用的小部件的构造函数。当指定一个父部件时，把它作为第二个参数进行传递，放在小部件名称后面，在小部件原型对象前面。

就像上面的实例，下面也要在 "custom" 命名空间中创建一个 "superDialog" 小部件。但是这次传递的是 jQuery UI 的 dialog（对话框）小部件 的构造函数（ `$.ui.dialog` ），表示 superDialog 小部件应该使用 jQuery UI 的 dialog（对话框）小部件作为父部件。

```
$.widget( "custom.superDialog", $.ui.dialog, {} );
```

在这里，superDialog 和 dialog 两个小部件实质上是等价的，只是名称和命名空间不同而已。为了让我们新的小部件更具特点，我们可以添加一些方法到它的原型对象上。

小部件的原型对象是传递给 `$.widget()` 的最后一个参数。到目前为止，我们的实例使用的是一个空的对象。现在让我们给这个对象添加一个方法：

```
$.widget( "custom.superDialog", $.ui.dialog, {
    red: function() {
        this.element.css( "color", "red" );
    }
});

// Create a new <div>, convert it into a superDialog, and call the
$( "<div>I am red</div>" )
    .superDialog()
    .superDialog( "red" );
```

现在 `superDialog` 有一个 `red()` 方法，这会把它的文本颜色改为红色。请注意，部件库（Widget Factory）是如何自动设置 `this` 为小部件的实例对象。如需了解实例上所有可用的方法和属性列表，请访问 部件库（Widget Factory） API 文档。

扩展已有的方法

有时候，您需要调整或添加已有部件方法的行为。您可以把方法名称指定为原型对象上需要重载的方法名称。下面的实例重载了 dialog（对话框）的 `open()` 方法。由于对话框默认是打开的，当运行这段代码时， `"open"` 将会被记录。

```
$.widget( "custom.superDialog", $.ui.dialog, {
    open: function() {
        console.log( "open" );
    }
});

// Create a new <div>, and convert it into a superDialog.
$( "<div>" ).superDialog();
```

当运行这段代码时，有一个问题。由于我们重载了 `open()` 的默认行为，所以 dialog（对话框）不再显示在屏幕上。

当我们在原型对象上使用方法，我们实际上是重载了原始的方法，在原型链中使用了一个新的方法。

为了让父部件方法可用，部件库（Widget Factory）提供了两个方法 - `_super()` 和 `_superApply()` 。

使用 `_super()` 和 `_superApply()` 来访问父部件

`_super()` 和 `_superApply()` 在父部件中调用了同样的方法。请看下面的实例。就像上一个实例，这个实例也重载了 `open()` 方法来记录 `"open"` 。然而，这次运行 `_super()` 是调用了 dialog（对话框）的 `open()` ，并打开对话框。

```
$.widget( "custom.superDialog", $.ui.dialog, {
    open: function() {
        console.log( "open" );

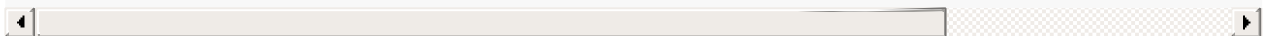
        // Invoke the parent widget's open().
        return this._super();
    }
});

$( "<div>" ).superDialog();
```

`_super()` 和 `_superApply()` 实际上等同于最初的 `Function.prototype.call()` 和 `Function.prototype.apply()` 方法。因此，`_super()` 接受一个参数列表，`_superApply()` 接受一个数组作为参数。下面的实例演示了这二者之间的不同。

```
$.widget( "custom.superDialog", $.ui.dialog, {
    _setOption: function( key, value ) {

        // Both invoke dialog's setOption() method. _super() requi
        // be passed as an argument list, _superApply() as a single
        this._super( key, value );
        this._superApply( arguments );
    }
});
```



重定义小部件

jQuery UI 1.9 添加了重定义小部件的功能。因此，可以不用创建一个新的小部件，我们只需要传递 `$.widget()` 这样一个已有的小部件名称和构造函数即可。下面的实例在 `open()` 中添加了相同的记录，但不是通过创建一个新的小部件来完成的。

```
$.widget( "ui.dialog", $.ui.dialog, {
    open: function() {
        console.log( "open" );
        return this._super();
    }
});

$( "<div>" ).dialog();
```

通过这个方法，我们可以扩展一个已有的小部件方法，但是仍然可以使用 `_super()` 来访问原始的方法 - 这些都不是通过创建一个新的小部件来完成的，而是直接重定义小部件即可。

小部件（Widgets）和多态性（Polymorphism）

当在小部件扩展及它们的插件之间进行交互时候，有一点值得注意，父部件的插件不能用来调用子部件元素上的方法。下面的实例演示了这一点。

```
$.widget( "custom.superDialog", $.ui.dialog, {} );

var dialog = $( "<div>" ).superDialog();

// This works.
dialog.superDialog( "close" );

// This doesn't.
dialog.dialog( "close" );
```

上面的实例中，父部件的插件，`dialog()`，不能调用 superDialog 元素上的 `close()` 方法。如需了解更多调用小部件方法的知识，请查看 小部件（Widget）方法调用。

定制个性化实例

目前为止，我们看到的实例都有在小部件原型上扩展的方法。在原型上重载的方法影响了小部件的所有实例。

为了演示这一点，请看下面的实例。dialog（对话框）的两个势力都使用了相同的 `open()` 方法。

```
$.widget( "ui.dialog", $.ui.dialog, {
    open: function() {
        console.log( "open" );
        return this._super();
    }
});

// Create two dialogs, both use the same open(), therefore "open" i
$( "<div>" ).dialog();
$( "<div>" ).dialog();
```

有时候，您只需要改变小部件的某个实例的行为。为了做到这点，您需要使用正常的 JavaScript 属性分配，获得对实例的引用，并重载该方法。具体如下面实例所示。

```
var dialogInstance = $( "<div>" )
    .dialog()
    // Retrieve the dialog's instance and store it.
    .data( "ui-dialog" );

// Override the close() method for this dialog
dialogInstance.close = function() {
    console.log( "close" );
};

// Create a second dialog
$( "<div>" ).dialog();

// Select both dialogs and call close() on each of them.
// "close" will only be logged once.
$( ":data(ui-dialog)" ).dialog( "close" );
```

个性化实例的重载方法技术是完美的一次性定制。

jQuery UI 小部件（Widget）方法调用

小部件（Widget）是通过 部件库（Widget Factory） 使用方法来改变他们初始化后的状态和执行动作而被创建的。有两种调用小部件方法的方式 - 通过部件库（Widget Factory）创建的插件，或者通过调用元素实例对象上的方法。

插件调用

使用小部件的插件调用方法，把方法名称以字符串形式进行传递。例如，点击这里查看，如何调用 dialog（对话框）小部件的 `close()` 方法。

```
$( ".selector" ).dialog( "close" );
```

如果方法要求参数，请作为额外的参数传递给插件。点击这里查看，如何调用 dialog（对话框）的 `option()` 方法。

```
$( ".selector" ).dialog( "option", "height" );
```

这会返回 dialog（对话框）的 `height` 选项 的值。

实例调用

每个小部件的每个实例都是使用 `jQuery.data()` 存储在元素上。为了检索实例对象，请使用小部件的全称作为键名调用 `jQuery.data()` 。具体如下面实例所示。

```
var dialog = $( ".selector" ).data( "ui-dialog" );
```

在您引用实例对象之后，可以直接在上面调用方法。

```
var dialog = $( ".selector" ).data( "ui-dialog" );
dialog.close();
```

在 jQuery UI 1.11 中，新的 `instance()` 方法会使得这个过程变得更简单。

```
$( ".selector" ).dialog( "instance" ).close();
```

返回类型

大多数通过小部件的插件调用的方法将返回一个 `jQuery` 对象，所以方法调用可以通过额外的 jQuery 方法链接。当在实例上进行调用时，则会返回 `undefined` 。具体如下面实例所示。

```
var dialog = $( ".selector" ).dialog();

// Instance invocation - returns undefined
dialog.data( "ui-dialog" ).close();

// Plugin invocation - returns a jQuery object
dialog.dialog( "close" );

// Therefore, plugin method invocation makes it possible to
// chain method calls with other jQuery functions
dialog.dialog( "close" )
    .css( "color", "red" );
```

例外的是，返回小部件相关信息的那些方法。例如 dialog（对话框）的 `isOpen()` 方法。

```
$( ".selector" )
    .dialog( "isOpen" )
    // This will throw a TypeError
    .css( "color", "red" );
```

这会产生一个 `TypeError` 错误，因为 `isOpen()` 返回的是一个布尔值，而不是一个 jQuery 对象。

jQuery UI 为什么使用部件库（Widget Factory）

编写 jQuery 插件与向 `jQuery.prototype` （通常显示为 `$.fn` ）添加方法一样简单，且需要遵循一些简单的规则，比如返回 `this` 。所以为什么会存在部件库（Widget Factory）？

在本章节中，我们将讲解部件库（Widget Factory）的好处，并了解何时使用它，以及为什么要使用它。

无状态 vs. 有状态插件

大多数 jQuery 插件是无状态的，它们执行一些动作即完成了它们的任务。例如，如果您使用 `.text( "hello" )` 设置元素的文本，没有安装阶段，结果都是一样的。对于这种类型的插件，它只是扩展了 jQuery 的原型。

然而，一些插件是有状态的，它们有全生命周期、维持状态以及对变化的反应。这些插件需要大量专门的代码来初始化和状态管理（有时是销毁）。这就导致出现了用于创建有状态插件的模板。更糟糕的是，每个插件的作者按照不同的方式进行管理插件的生命周期和状态，这就导致了不同的插件有不同的 API 样式。部件库（Widget Factory）旨在解决这些问题，它移除了模板，并为插件创建了一个一致的 API。

一致的 API

部件库（Widget Factory）定义了如何创建和销毁小部件，获取和设置选项，调用方法，以及监听小部件触发的事件。通过使用部件库（Widget Factory）来创建有状态的插件，会自动符合定义的标准，让新用户更容易使用您的插件。另外，部件库（Widget Factory）还能实现定义接口的功能。如果您对部件库（Widget Factory）提供的 API 还不熟悉，请查看 如何使用部件库（Widget Factory）。

在初始化时设置选项

当您创建一个接受选项的插件时，您应该为尽可能多的选项定义 defaults。然后在初始化时，把用户提供的选项与 defaults 进行合并。您也可以暴露 defaults，这样用户就可以更改默认值。在 jQuery 插件中，一个常用的模式如下所示：

```
$.fn.plugin = function( options ) {
    options = $.extend( {}, $.fn.plugin.defaults, options );
    // Plugin logic goes here.
};

$.fn.plugin.defaults = {
    param1: "foo",
    param2: "bar",
    param3: "baz"
};
```

部件库（Widget Factory）也提供了这个功能，并在这上面做了改进。使用部件库（Widget Factory）之后，它将如下所示。

```
$.widget( "ns.plugin", {

    // Default options.
    options: {
        param1: "foo",
        param2: "bar",
        param3: "baz"
    },

    _create: function() {
        // Options are already merged and stored in this.options
        // Plugin logic goes here.
    }

});
```

jQuery UI 如何使用部件库（Widget Factory）

我们将创建一个进度条。正如下面实例所示，这可以通过调用 `jQuery.widget()` 来完成，它带有两个参数：一个是要创建的插件名称，一个是包含支持插件的函数的对象文字。当插件被调用时，它将创建一个新的插件实例，所有的函数都将在该实例的语境中被执行。这与两种重要方式的标准 jQuery 插件不同。首先，语境是一个对象，不是 DOM 元素。其次，语境总是一个单一的对象，不是一个集合。

```
$.widget( "custom.progressbar", {
    _create: function() {
        var progress = this.options.value + "%";
        this.element
            .addClass( "progressbar" )
            .text( progress );
    }
});
```

插件的名称必须包含命名空间，在这个实例中，我们使用了 `custom` 命名空间。您只能创建一层深的命名空间，因此， `custom.progressbar` 是一个有效的插件名称，而 `very.custom.progressbar` 不是一个有效的插件名称。

我们看到部件库（Widget Factory）为我们提供了两个属性。 `this.element` 是一个包含一个元素的 jQuery 对象。如果我们的插件在包含多个元素的 jQuery 对象上调用，则会为每个元素创建一个单独的插件实例，且每个实例都会有自己的 `this.element` 。第二个属性， `this.options` ，是一个包含所有插件选项的键名/键值对的哈希（hash）。这些选项可以被传给插件，如下所示：

```
$( "<div></div>" )
    .appendTo( "body" )
    .progressbar({ value: 20 });
```

当我们调用 `jQuery.widget()` ，它通过给 `jQuery.fn` （用于创建标准插件的系统） 添加函数来扩展 jQuery。所添加的函数名称是基于您传给 `jQuery.widget()` 的名称，不带命名空间 - "progressbar"。传给插件的选项是在插件实例中获取设置的值。如下面的实例所示，我们可以为任意一个选项指定默认值。当设计您的 API 时，您应该清楚你的插件的最常见的使用情况，以便您可以设置适当的默认值，且确保使所有的选项真正可选。

```
$.widget( "custom.progressbar", {

    // Default options.
    options: {
        value: 0
    },
    _create: function() {
        var progress = this.options.value + "%";
        this.element
            .addClass( "progressbar" )
            .text( progress );
    }
});
```

调用插件方法

现在我们可以初始化我们的进度条，我们将通过在插件实例上调用方法来执行动作。为了定义一个插件方法，我们只在我们传给 `jQuery.widget()` 的对象中引用函数。我们也可以通过给函数名加下划线前缀来定义 "private" 方法。

```
$.widget( "custom.progressbar", {

    options: {
        value: 0
    },

    _create: function() {
        var progress = this.options.value + "%";
        this.element
            .addClass( "progressbar" )
            .text( progress );
    },

    // Create a public method.
    value: function( value ) {

        // No value passed, act as a getter.
        if ( value === undefined ) {
            return this.options.value;
        }

        // Value passed, act as a setter.
        this.options.value = this._constrain( value );
        var progress = this.options.value + "%";
        this.element.text( progress );
    },

    // Create a private method.
    _constrain: function( value ) {
        if ( value > 100 ) {
            value = 100;
        }
        if ( value < 0 ) {
            value = 0;
        }
        return value;
    }
});
```

为了在插件实例上调用方法，您可以向 jQuery 插件传递方法的名称。如果您调用的方法接受参数，您只需简单地在方法名后面传递这些参数即可。

注意：通过向同一个用于初始化插件的 jQuery 函数传递方法名来执行方法。这样做是为了在保持链方法调用时防止 jQuery 命名空间污染。在本章节的后续，我们将看到看起来更自然的其他用法。

```
var bar = $( "<div></div>" )
    .appendTo( "body" )
    .progressbar({ value: 20 });

// Get the current value.
alert( bar.progressbar( "value" ) );

// Update the value.
bar.progressbar( "value", 50 );

// Get the current value again.
alert( bar.progressbar( "value" ) );
```

使用选项

`option()` 方法是自动提供给插件的。 `option()` 方法允许您在初始化后获取并设置选项。该方法像 jQuery 的 `.css()` 和 `.attr()` 方法：您可以只传递一个名称作为取值器来使用，也可以传递一个名称和值作为设置器使用，或者传递一个键名/键值对的哈希来设置多个值。当作为取值器使用时，插件将返回与传入名称相对应的选项的当前值。当作为设置器使用时，插件的 `_setOption` 方法将被每个被设置的选项调用。我们可以在我们的插件中指定一个 `_setOption` 方法来反应选项更改。对于更改选项要独立执行的动作，我们可以重载 `_setOptions` 。

```
$.widget( "custom.progressbar", {
    options: {
        value: 0
    },
    _create: function() {
        this.options.value = this._constrain(this.options.value);
        this.element.addClass( "progressbar" );
        this.refresh();
    },
    _setOption: function( key, value ) {
        if ( key === "value" ) {
            value = this._constrain( value );
        }
        this._super( key, value );
    },
    _setOptions: function( options ) {
        this._super( options );
        this.refresh();
    },
    refresh: function() {
        var progress = this.options.value + "%";
        this.element.text( progress );
    },
    _constrain: function( value ) {
        if ( value > 100 ) {
            value = 100;
        }
        if ( value < 0 ) {
            value = 0;
        }
        return value;
    }
});
```

添加回调

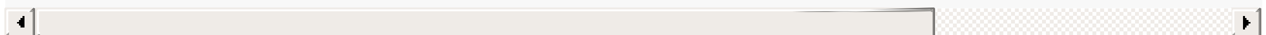
最简单的扩展插件的方法是添加回调，这样用户就可以在插件状态发生变化时做出反应。我们可以看下面的实例如何在进度达到 100% 时添加回调到进度条。 `_trigger()` 方法有三个参数：回调名称，一个启动回调的 jQuery 事件对象，以及一个与事件相关的数据哈希。回调名称是唯一一个必需的参数，但是对于想要在插件上实现自定义功能的用户，其他的参数是非常有用的。例如，如果我们创建一个可拖拽插件，我们可以在触发拖拽回调时传递 mousemove 事件，这将允许用户对基于由事件对象提供的 x/y 坐标上的拖拽做出反应。请注意，传递到 `_trigger()` 的原始的事件必须是一个 jQuery 事件，而不是一个原生的浏览器事件。

```
$.widget( "custom.progressbar", {
    options: {
        value: 0
    },
    _create: function() {
        this.options.value = this._constrain(this.options.value);
        this.element.addClass( "progressbar" );
        this.refresh();
    },
    _setOption: function( key, value ) {
        if ( key === "value" ) {
            value = this._constrain( value );
        }
        this._super( key, value );
    },
    _setOptions: function( options ) {
        this._super( options );
        this.refresh();
    },
    refresh: function() {
        var progress = this.options.value + "%";
        this.element.text( progress );
        if ( this.options.value == 100 ) {
            this._trigger( "complete", null, { value: 100 } );
        }
    },
    _constrain: function( value ) {
        if ( value > 100 ) {
            value = 100;
        }
        if ( value < 0 ) {
            value = 0;
        }
        return value;
    }
});
```

回调函数本质上只是附加选项，所以您可以像其他选项一样获取并设置它们。无论何时执行回调，都会有一个相对应的事件被触发。事件类型是通过连接插件的名称和回调函数名称确定的。回调和事件都接受两个相同的参数：一个事件对象和一个与事件相关的数据哈希，具体如下面实例所示。您的插件可能需要包含防止用户使用的功能，为了做到这点，最好的方法就是创建爱你可撤销的回调。用户可以撤销回调或者相关的事件，与他们撤销任何一个原生事件一样，都是通过调用 `event.preventDefault()` 或返回 `false` 来实现的。如果用户撤销回调， `_trigger()` 方法将返回 `false` ，这样您就能在插件内实现合适的功能。

```
var bar = $( "<div></div>" )
    .appendTo( "body" )
    .progressbar({
        complete: function( event, data ) {
            alert( "Callbacks are great!" );
        }
    })
    .bind( "progressbarcomplete", function( event, data ) {
        alert( "Events bubble and support many handlers for extreme
        alert( "The progress bar value is " + data.value );
    });

bar.progressbar( "option", "value", 100 );
```



本质

现在我们已经看到如何使用部件库（Widget Factory）创建一个插件，接下来让我们看看它实际上是如何工作的。当您调用 `jQuery.widget()` 时，它将为插件创建一个构造函数，并设置您为插件实例传入的作为原型的对象。所有自动添加到插件的功能都来自一个基本的小部件原型，该原型定义为 `jQuery.Widget.prototype`。当创建插件实例时，会使用 `jQuery.data` 把它存储在原始的 DOM 元素上，插件名作为键名。

由于插件实例直接链接到 DOM 元素上，您可以直接访问插件实例，而不需要遍历插件方法。这将允许您直接在插件实例上调用方法，而不需要传递字符串形式的方法名，同时您也可以直接访问插件的属性。

```
var bar = $( "<div></div>" )
    .appendTo( "body" )
    .progressbar()
    .data( "progressbar" );

// Call a method directly on the plugin instance.
bar.option( "value", 50 );

// Access properties on the plugin instance.
alert( bar.options.value );
```

您也可以在不遍历插件方法的情况下创建一个实例，通过选项和元素直接调用构造函数即可：

```
var bar = $.custom.progressbar( {}, $( "<div></div>" ).appendTo( "b

// Same result as before.
alert( bar.options.value );
```

扩展插件的原型

插件有构造函数和原型的最大好处是易于扩展插件。通过添加或修改插件原型上的方法，我们可以修改插件所有实例的行为。例如，如果我们想要向进度条添加一个方法来重置进度为 0%，我们可以向原型添加这个方法，它将在所有插件实例上可调用。

```
$.custom.progressbar.prototype.reset = function() {
    this._setOption( "value", 0 );
};
```

如需了解扩展小部件的更多细节，以及如何在一个已有的小部件上创建一个全新的小部件的更多细节，请查看 通过部件库（Widget Factory）扩展小部件（Widget）。

清理

在某些情况下，允许用户应用插件，然后再取消应用。您可以通过 `_destroy()` 方法做到这一点。在 `_destroy()` 方法内，您应该撤销在初始化和后期使用期间插件所做的一切动作。 `_destroy()` 是通过 `.destroy()` 方法被调用的， `.destroy()` 方法是在插件实例绑定的元素从 DOM 上移除时被自动调用的，所以这可被用于垃圾回收。基本的 `.destroy()` 方法也处理一些常用的清理操作，比如从小部件的 DOM 元素上移除实例引用，从元素上解除绑定小部件命名空间中的所有事件，解除绑定所有使用 `_bind()` 添加的事件。

```
$.widget( "custom.progressbar", {
    options: {
        value: 0
    },
    _create: function() {
        this.options.value = this._constrain(this.options.value);
        this.element.addClass( "progressbar" );
        this.refresh();
    },
    _setOption: function( key, value ) {
        if ( key === "value" ) {
            value = this._constrain( value );
        }
        this._super( key, value );
    },
    _setOptions: function( options ) {
        this._super( options );
        this.refresh();
    },
    refresh: function() {
        var progress = this.options.value + "%";
        this.element.text( progress );
        if ( this.options.value == 100 ) {
            this._trigger( "complete", null, { value: 100 } );
        }
    },
    _constrain: function( value ) {
        if ( value > 100 ) {
            value = 100;
        }
        if ( value < 0 ) {
            value = 0;
        }
        return value;
    },
    _destroy: function() {
        this.element
            .removeClass( "progressbar" )
            .text( "" );
    }
});
```

关闭注释

部件库（Widget Factory）只是创建有状态插件的一种方式。这里还有一些其他不同的模型可以使用，且每个都有各自的优势和劣势。部件库（Widget Factory）解决了很多常见的问题，且大大提高了效率，同时也大大提高了代码的重用性，使它适合于 jQuery UI 及其他有状态的插件。

请注意，在本章节中我们使用了 `custom` 命名空间。 `ui` 命名空间被官方的 jQuery UI 插件保留。当创建您自己的插件时，您应该创建自己的命名空间。这样才能更清楚插件来自哪里，属于哪个范围。

jQuery UI 实例

jQuery UI 实例

jQuery UI 提供了一组用户界面交互、特效、小部件、实用工具及主题。通过实例演示、查看源代码、创建主题、阅读API 文档，开始学习使用 jQuery UI。

交互（Interactions）

交互（Interactions）向任何元素添加基本的基于鼠标的行为。您可以只通过短短的几行代码即可创建排序列表、缩放元素、拖放行为等等。交互（Interactions）也作用于更复杂的小部件和应用程序。

- 拖动（Draggable）
- 放置（Droppable）
- 缩放（Resizable）
- 选择（Selectable）
- 排序（Sortable）

小部件（Widgets）

小部件（Widgets）有功能齐全的 UI 控件，使桌面应用程序也具备 Web 应用程序一样丰富的功能。所有的小部件（Widgets）提供了一个核心，带有定制行为的大量扩展以及完整的主题支持。

- 折叠面板（Accordion）
- 自动完成（Autocomplete）
- 按钮（Button）
- 日期选择器（Datepicker）
- 对话框（Dialog）
- 菜单（Menu）
- 进度条（Progressbar）
- 滑块（Slider）
- 旋转器（Spinner）
- 标签页（Tabs）
- 工具提示框（Tooltip）

特效（Effects）

特效（Effects）支持颜色动画和 Class 转换，同时也提供了一些额外的 Easings。另外，提供了一套完整的定制特效，供显示和隐藏元素时或者只是添加一些视觉显示时使用。

- 特效（Effect）
- 可见性（Visibility）

    - 显示（Show）
    - 隐藏（Hide）
    - 切换（Toggle）
- Class 动画（Class Animation）
    - 添加 Class（Add Class）
    - 移除 Class（Remove Class）
    - 切换 Class（Toggle Class）
    - 转换 Class（Switch Class）
- 颜色动画（Color Animation）

实用工具（Utilities）

jQuery UI 使用实体工具（Utilities）来创建交互（interactions）和小部件（widgets）。

- 定位（Position）
- 部件库（Widget Factory）

jQuery UI 实例 - 拖动（Draggable）

允许使用鼠标移动元素。

如需了解更多有关 draggable 交互的细节，请查看 API 文档 可拖拽小部件（Draggable Widget）。

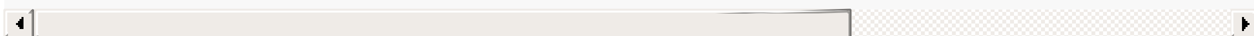
默认功能

在任意的 DOM 元素上启用 draggable 功能。通过鼠标点击并在视区中拖动来移动 draggable 对象。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 拖动（Draggable） - 默认功能</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #draggable { width: 150px; height: 150px; padding: 0.5em; }
  </style>
  <script>
  $(function() {
    $( "#draggable" ).draggable();
  });
  </script>
</head>
<body>

<div id="draggable" class="ui-widget-content">
  <p>请拖动我！</p>
</div>

</body>
</html>
```



自动滚动

当 draggable 移动到视区外时自动滚动文档。设置 `scroll` 选项为 true 来启用自动滚动，当滚动触发时进行微调，滚动速度是通过 `scrollSensitivity` 和 `scrollSpeed` 选项设置的。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 拖动（Draggable） - 自动滚动</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #draggable, #draggable2, #draggable3 { width: 100px; height: 100p
  </style>
  <script>
  $(function() {
    $( "#draggable" ).draggable({ scroll: true });
    $( "#draggable2" ).draggable({ scroll: true, scrollSensitivity
    $( "#draggable3" ).draggable({ scroll: true, scrollSpeed: 100 }
  });
  </script>
</head>
<body>

<div id="draggable" class="ui-widget-content">
  <p>Scroll 设置为 true，默认设置</p>
</div>

<div id="draggable2" class="ui-widget-content">
  <p>scrollSensitivity 设置为 100</p>
</div>

<div id="draggable3" class="ui-widget-content">
  <p>scrollSpeed 设置为 100</p>
</div>

<div style="height: 5000px; width: 1px;"></div>

</body>
</html>
```

约束运动

通过定义 draggable 区域的边界来约束每个 draggable 的运动。设置 `axis` 选项来限制 draggable 的路径为 x 轴或者 y 轴，或者使用 `containment` 选项来指定一个父级的 DOM 元素或者一个 jQuery 选择器，比如 'document.'。

```
<!doctype html>
<html lang="en">
<head>
```

```
  <meta charset="utf-8">
  <title>jQuery UI 拖动（Draggable） - 约束运动</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  .draggable { width: 90px; height: 90px; padding: 0.5em; float: le
  #draggable, #draggable2 { margin-bottom:20px; }
  #draggable { cursor: n-resize; }
  #draggable2 { cursor: e-resize; }
  #containment-wrapper { width: 95%; height:150px; border:2px solic
  h3 { clear: left; }
  </style>
  <script>
  $(function() {
    $( "#draggable" ).draggable({ axis: "y" });
    $( "#draggable2" ).draggable({ axis: "x" });

    $( "#draggable3" ).draggable({ containment: "#containment-wrapp
    $( "#draggable5" ).draggable({ containment: "parent" });
  });
  </script>
</head>
<body>

<h3>沿着轴约束运动：</h3>

<div id="draggable" class="draggable ui-widget-content">
  <p>只能垂直拖拽</p>
</div>

<div id="draggable2" class="draggable ui-widget-content">
  <p>只能水平拖拽</p>
</div>

<h3>或者在另一个 DOM 元素中约束运动：</h3>
<div id="containment-wrapper">
  <div id="draggable3" class="draggable ui-widget-content">
    <p>我被约束在盒子里</p>
  </div>

  <div class="draggable ui-widget-content">
    <p id="draggable5" class="ui-widget-header">我被约束在父元素内</p>
  </div>
</div>

</body>
</html>
```

光标样式

当拖拽对象时定位光标。默认情况下，光标是出现在被拖拽对象的中间。使用 `cursorAt` 选项来指定相对于 draggable 的另一个位置（指定一个相对于 top、right、bottom、left 的像素值）。通过提供一个带有有效的 CSS 光标值的 `cursor` 选项，来自定义光标的外观。有效的 CSS 光标值包括：default、move、pointer、crosshair，等等。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 拖动（Draggable） - 光标样式</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #draggable, #draggable2, #draggable3 { width: 100px; height: 100p
  </style>
  <script>
  $(function() {
    $( "#draggable" ).draggable({ cursor: "move", cursorAt: { top:
    $( "#draggable2" ).draggable({ cursor: "crosshair", cursorAt: {
    $( "#draggable3" ).draggable({ cursorAt: { bottom: 0 } });
  });
  </script>
</head>
<body>

<div id="draggable" class="ui-widget-content">
  <p>我总是在中间（相对于鼠标）</p>
</div>

<div id="draggable2" class="ui-widget-content">
  <p>我的光标是在 left -5 和 top -5</p>
</div>

<div id="draggable3" class="ui-widget-content">
  <p>我的光标位置只控制了 'bottom' 值</p>
</div>

</body>
</html>
```

延迟开始

通过 `delay` 选项设置延迟开始拖拽的毫秒数。通过 `distance` 选项设置光标被按下且拖拽指定像素后才允许拖拽。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 拖动（Draggable） - 延迟开始</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #draggable, #draggable2 { width: 120px; height: 120px; padding: 0
  </style>
  <script>
  $(function() {
    $( "#draggable" ).draggable({ distance: 20 });
    $( "#draggable2" ).draggable({ delay: 1000 });
    $( ".ui-draggable" ).disableSelection();
  });
  </script>
</head>
<body>

<div id="draggable" class="ui-widget-content">
  <p>只有把我拖拽了 20 像素后，拖拽才开始</p>
</div>

<div id="draggable2" class="ui-widget-content">
  <p>不管 distance 是多少，您都必须拖拽并等待 1000ms 后拖拽才开始</p>
</div>

</body>
</html>
```



事件

draggable 上的 `start` 、 `drag` 和 `stop` 事件。拖拽开始时触发 `start` 事件，拖拽期间触发 `drag` 事件，拖拽停止时触发 `stop` 事件。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 拖动（Draggable） - 事件</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
```

```
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #draggable { width: 16em; padding: 0 1em; }
  #draggable ul li { margin: 1em 0; padding: 0.5em 0; } * html #dra
  #draggable ul li span.ui-icon { float: left; }
  #draggable ul li span.count { font-weight: bold; }
  </style>
  <script>
  $(function() {
    var $start_counter = $( "#event-start" ),
      $drag_counter = $( "#event-drag" ),
      $stop_counter = $( "#event-stop" ),
      counts = [ 0, 0, 0 ];

    $( "#draggable" ).draggable({
      start: function() {
        counts[ 0 ]++;
        updateCounterStatus( $start_counter, counts[ 0 ] );
      },
      drag: function() {
        counts[ 1 ]++;
        updateCounterStatus( $drag_counter, counts[ 1 ] );
      },
      stop: function() {
        counts[ 2 ]++;
        updateCounterStatus( $stop_counter, counts[ 2 ] );
      }
    });

    function updateCounterStatus( $event_counter, new_count ) {
      // 首先更新视觉状态...
      if ( !$event_counter.hasClass( "ui-state-hover" ) ) {
        $event_counter.addClass( "ui-state-hover" )
          .siblings().removeClass( "ui-state-hover" );
      }
      // ...然后更新数字
      $( "span.count", $event_counter ).text( new_count );
    }
  });
  </script>
</head>
<body>

<div id="draggable" class="ui-widget ui-widget-content">

  <p>请拖拽我，触发一连串的事件。</p>

  <ul class="ui-helper-reset">
    <li id="event-start" class="ui-state-default ui-corner-all"><sp
    <li id="event-drag" class="ui-state-default ui-corner-all"><spa
    <li id="event-stop" class="ui-state-default ui-corner-all"><spa
  </ul>
```

```
</div>

</body>
</html>
```

Handles

只有当光标在 draggable 上指定部分时才允许拖拽。使用 `handle` 选项来指定用于拖拽对象的元素（或元素组）的 jQuery 选择器。

Or prevent dragging when the cursor is over a specific element (or group of elements) within the draggable. Use the cancel option to specify a jQuery selector over which to "cancel" draggable functionality.

或者当光标在 draggable 内指定元素（或元素组）上时不允许拖拽。使用 `handle` 选项来指定取消拖拽功能的 jQuery 选择器。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 拖动（Draggable） - Handles</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #draggable, #draggable2 { width: 100px; height: 100px; padding: 0
  #draggable p { cursor: move; }
  </style>
  <script>
  $(function() {
    $( "#draggable" ).draggable({ handle: "p" });
    $( "#draggable2" ).draggable({ cancel: "p.ui-widget-header" });
    $( "div, p" ).disableSelection();
  });
  </script>
</head>
<body>

<div id="draggable" class="ui-widget-content">
  <p class="ui-widget-header">您只可以在这个范围内拖拽我</p>
</div>

<div id="draggable2" class="ui-widget-content">
  <p>您可以把我向四周拖拽&hellip;</p>
  <p class="ui-widget-header">&hellip;但是您不可以在这个范围内拖拽我</p>
</div>

</body>
</html>
```

还原位置

当带有布尔值 `revert` 选项的 draggable 停止拖拽时，返回 draggable（或它的助手）到原始位置。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 拖动（Draggable） - 还原位置</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #draggable, #draggable2 { width: 100px; height: 100px; padding: (
  </style>
  <script>
  $(function() {
    $( "#draggable" ).draggable({ revert: true });
    $( "#draggable2" ).draggable({ revert: true, helper: "clone" })
  });
  </script>
</head>
<body>

<div id="draggable" class="ui-widget-content">
  <p>还原</p>
</div>

<div id="draggable2" class="ui-widget-content">
  <p>还原助手</p>
</div>

</body>
</html>
```

对齐到元素或网格

对齐 draggable 到 DOM 元素的内部或外部边界。使用 `snap` 、 `snapMode` （inner, outer, both）和 `snapTolerance` （当调用对齐时，draggable 与元素之间的距离，以像素为单位）选项。

或者对齐 draggable 到网格。通过 `grid` 选项设置网格单元的尺寸（以像素为单位的高度或宽度）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 拖动（Draggable） - 对齐到元素或网格</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
```

```
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  .draggable { width: 90px; height: 80px; padding: 5px; float: left
  .ui-widget-header p, .ui-widget-content p { margin: 0; }
  #snaptarget { height: 140px; }
  </style>
  <script>
  $(function() {
    $( "#draggable" ).draggable({ snap: true });
    $( "#draggable2" ).draggable({ snap: ".ui-widget-header" });
    $( "#draggable3" ).draggable({ snap: ".ui-widget-header", snapM
    $( "#draggable4" ).draggable({ grid: [ 20, 20 ] });
    $( "#draggable5" ).draggable({ grid: [ 80, 80 ] });
  });
  </script>
</head>
<body>

<div id="snaptarget" class="ui-widget-header">
  <p>我是对齐目标</p>
</div>

<br style="clear:both">

<div id="draggable" class="draggable ui-widget-content">
  <p>默认（snap: true），对齐到所有其他的 draggable 元素</p>
</div>

<div id="draggable2" class="draggable ui-widget-content">
  <p>我只对齐到大盒子</p>
</div>

<div id="draggable3" class="draggable ui-widget-content">
  <p>我只对齐到大盒子的外边缘</p>
</div>

<div id="draggable4" class="draggable ui-widget-content">
  <p>我对齐到一个 20 x 20 网格</p>
</div>

<div id="draggable5" class="draggable ui-widget-content">
  <p>我对齐到一个 80 x 80 网格</p>
</div>

</body>
</html>
```

视觉反馈

给用户提供反馈，就像以助手方式拖拽对象一样。 `helper` 选项接受值 'original'（用光标移动 draggable 对象），'clone'（用光标移动 draggable 的副本），或者一个返回 DOM 元素的函数（该元素在拖拽期间显示在光标附近）。通过 `opacity` 选项控制助手的透明度。

为了区别哪一个 draggable 正在被拖拽，让在运动中的 draggable 保持最前。如果正在拖拽，使用 `zIndex` 选项来设置助手的高度 z-index，或者使用 `stack` 选项来确保当停止拖拽时，最后一个被拖拽的项目总是出现在同组其他项目的上面。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 拖动（Draggable） - 视觉反馈</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #draggable, #draggable2, #draggable3, #set div { width: 90px; hei
  #draggable, #draggable2, #draggable3 { margin-bottom:20px; }
  #set { clear:both; float:left; width: 368px; height: 120px; }
  p { clear:both; margin:0; padding:1em 0; }
  </style>
  <script>
  $(function() {
    $( "#draggable" ).draggable({ helper: "original" });
    $( "#draggable2" ).draggable({ opacity: 0.7, helper: "clone" })
    $( "#draggable3" ).draggable({
      cursor: "move",
      cursorAt: { top: -12, left: -20 },
      helper: function( event ) {
        return $( "<div class='ui-widget-header'>I'm a custom helpe
      }
    });
    $( "#set div" ).draggable({ stack: "#set div" });
  });
  </script>
</head>
<body>

<h3 class="docs">助手：</h3>

<div id="draggable" class="ui-widget-content">
  <p>原始的</p>
</div>

<div id="draggable2" class="ui-widget-content">
  <p>半透明的克隆</p>
</div>

<div id="draggable3" class="ui-widget-content">
```

```
  <p>自定义助手（与 cursorAt 结合）</p>
</div>

<h3 class="docs">堆叠：</h3>
<div id="set">
  <div class="ui-widget-content">
    <p>我们是 draggables..</p>
  </div>

  <div class="ui-widget-content">
    <p>..它的 z-index 是自动控制的..</p>
  </div>

  <div class="ui-widget-content">
    <p>..带有 stack 选项。</p>
  </div>
</div>

</body>
</html>
```

jQuery UI Draggable + Sortable

Draggable 与 Sortable 的无缝交互。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 拖动（Draggable） + 排序（Sortable）</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  ul { list-style-type: none; margin: 0; padding: 0; margin-bottom
  li { margin: 5px; padding: 5px; width: 150px; }
  </style>
  <script>
  $(function() {
    $( "#sortable" ).sortable({
      revert: true
    });
    $( "#draggable" ).draggable({
      connectToSortable: "#sortable",
      helper: "clone",
      revert: "invalid"
    });
    $( "ul, li" ).disableSelection();
  });
  </script>
</head>
<body>

<ul>
  <li id="draggable" class="ui-state-highlight">请拖拽我</li>
</ul>

<ul id="sortable">
  <li class="ui-state-default">Item 1</li>
  <li class="ui-state-default">Item 2</li>
  <li class="ui-state-default">Item 3</li>
  <li class="ui-state-default">Item 4</li>
  <li class="ui-state-default">Item 5</li>
</ul>

</body>
</html>
```

jQuery UI 实例 - 放置（Droppable）

为可拖拽小部件创建目标。

如需了解更多有关 droppable 交互的细节，请查看 API 文档 可放置小部件（Droppable Widget）。

默认功能

在任意的 DOM 元素上启用 droppable 功能，并为可拖拽小部件创建目标。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 放置（Droppable） - 默认功能</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #draggable { width: 100px; height: 100px; padding: 0.5em; float:
  #droppable { width: 150px; height: 150px; padding: 0.5em; float:
  </style>
  <script>
  $(function() {
    $( "#draggable" ).draggable();
    $( "#droppable" ).droppable({
      drop: function( event, ui ) {
        $( this )
          .addClass( "ui-state-highlight" )
          .find( "p" )
            .html( "Dropped!" );
      }
    });
  });
  </script>
</head>
<body>

<div id="draggable" class="ui-widget-content">
  <p>请把我拖拽到目标处！</p>
</div>

<div id="droppable" class="ui-widget-header">
  <p>请放置在这里！</p>
</div>

</body>
</html>
```

接受

使用 `accept` 选项指定目标 droppable 接受哪一个元素（或元素组）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 放置（Droppable） - 接受</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #droppable { width: 150px; height: 150px; padding: 0.5em; float: 
  #draggable, #draggable-nonvalid { width: 100px; height: 100px; pa
  </style>
  <script>
  $(function() {
    $( "#draggable, #draggable-nonvalid" ).draggable();
    $( "#droppable" ).droppable({
      accept: "#draggable",
      activeClass: "ui-state-hover",
      hoverClass: "ui-state-active",
      drop: function( event, ui ) {
        $( this )
          .addClass( "ui-state-highlight" )
          .find( "p" )
            .html( "Dropped!" );
      }
    });
  });
  </script>
</head>
<body>

<div id="draggable-nonvalid" class="ui-widget-content">
  <p>我是不能被放置的 draggable</p>
</div>

<div id="draggable" class="ui-widget-content">
  <p>请拖拽我到目标</p>
</div>

<div id="droppable" class="ui-widget-header">
  <p>accept: '#draggable'</p>
</div>

</body>
</html>
```

防止传播

当使用嵌套的 droppable 时 — 例如，您可以有一个树形的可编辑的目录结构，带有文件夹和文档节点 — `greedy` 选项设置为 true 来防止当 draggable 被放置在子节点（droppable）上时的事件传播。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 放置（Droppable） - 防止传播</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #draggable { width: 100px; height: 40px; padding: 0.5em; float: l
  #droppable, #droppable2 { width: 230px; height: 120px; padding: 0
  #droppable-inner, #droppable2-inner { width: 170px; height: 60px;
  </style>
  <script>
  $(function() {
    $( "#draggable" ).draggable();

    $( "#droppable, #droppable-inner" ).droppable({
      activeClass: "ui-state-hover",
      hoverClass: "ui-state-active",
      drop: function( event, ui ) {
        $( this )
          .addClass( "ui-state-highlight" )
          .find( "> p" )
            .html( "Dropped!" );
        return false;
      }
    });

    $( "#droppable2, #droppable2-inner" ).droppable({
      greedy: true,
      activeClass: "ui-state-hover",
      hoverClass: "ui-state-active",
      drop: function( event, ui ) {
        $( this )
          .addClass( "ui-state-highlight" )
          .find( "> p" )
            .html( "Dropped!" );
      }
    });
  });
  </script>
</head>
<body>

<div id="draggable" class="ui-widget-content">
  <p>请拖拽我到目标</p>
```

```
</div>

<div id="droppable" class="ui-widget-header">
  <p>外部 droppable</p>
  <div id="droppable-inner" class="ui-widget-header">
    <p>内部 droppable（不带有 greedy）</p>
  </div>
</div>

<div id="droppable2" class="ui-widget-header">
  <p>外部 droppable</p>
  <div id="droppable2-inner" class="ui-widget-header">
    <p>内部 droppable（带有 greedy）</p>
  </div>
</div>

</body>
</html>
```

还原 draggable 的位置

当带有布尔值 `revert` 选项的 draggable 停止拖拽时，返回 draggable（或它的助手）到原始位置。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 放置（Droppable） - 还原 draggable 的位置</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #draggable, #draggable2 { width: 100px; height: 100px; padding: 0
  #droppable { width: 150px; height: 150px; padding: 0.5em; float: 
  </style>
  <script>
  $(function() {
    $( "#draggable" ).draggable({ revert: "valid" });
    $( "#draggable2" ).draggable({ revert: "invalid" });

    $( "#droppable" ).droppable({
      activeClass: "ui-state-default",
      hoverClass: "ui-state-hover",
      drop: function( event, ui ) {
        $( this )
          .addClass( "ui-state-highlight" )
          .find( "p" )
            .html( "已放置！" );
      }
    });
  });
  </script>
</head>
<body>

<div id="draggable" class="ui-widget-content">
  <p>当被放置在目标上时还原</p>
</div>

<div id="draggable2" class="ui-widget-content">
  <p>当未被放置在目标上时还原</p>
</div>

<div id="droppable" class="ui-widget-header">
  <p>请放置在这里</p>
</div>

</body>
</html>
```

购物车演示

演示如何使用折叠面板来展示产品的目录结构，使用拖拽和放置来添加产品到购物车，购物车中的产品是可排序的。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 放置（Droppable） - 购物车演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  h1 { padding: .2em; margin: 0; }
  #products { float:left; width: 500px; margin-right: 2em; }
  #cart { width: 200px; float: left; margin-top: 1em; }
  /* 定义列表样式，以便最大化 droppable */
  #cart ol { margin: 0; padding: 1em 0 1em 3em; }
  </style>
  <script>
  $(function() {
    $( "#catalog" ).accordion();
    $( "#catalog li" ).draggable({
      appendTo: "body",
      helper: "clone"
    });
    $( "#cart ol" ).droppable({
      activeClass: "ui-state-default",
      hoverClass: "ui-state-hover",
      accept: ":not(.ui-sortable-helper)",
      drop: function( event, ui ) {
        $( this ).find( ".placeholder" ).remove();
        $( "<li></li>" ).text( ui.draggable.text() ).appendTo( this
      }
    }).sortable({
      items: "li:not(.placeholder)",
      sort: function() {
        // 获取由 droppable 与 sortable 交互而加入的条目
        // 使用 connectWithSortable 可以解决这个问题，但不允许您自定义 ac
        $( this ).removeClass( "ui-state-default" );
      }
    });
  });
  </script>
</head>
<body>

<div id="products">
  <h1 class="ui-widget-header">产品</h1>
  <div id="catalog">
    <h2><a href="#">T-Shirts</a></h2>
    <div>
```

```
      <ul>
        <li>Lolcat Shirt</li>
        <li>Cheezeburger Shirt</li>
        <li>Buckit Shirt</li>
      </ul>
    </div>
    <h2><a href="#">Bags</a></h2>
    <div>
      <ul>
        <li>Zebra Striped</li>
        <li>Black Leather</li>
        <li>Alligator Leather</li>
      </ul>
    </div>
    <h2><a href="#">Gadgets</a></h2>
    <div>
      <ul>
        <li>iPhone</li>
        <li>iPod</li>
        <li>iPad</li>
      </ul>
    </div>
  </div>
</div>

<div id="cart">
  <h1 class="ui-widget-header">购物车</h1>
  <div class="ui-widget-content">
    <ol>
      <li class="placeholder">添加产品到这里</li>
    </ol>
  </div>
</div>

</body>
</html>
```

简单的照片管理器

您可以通过拖拽照片到回收站或者点击回收站图标来删除照片。

您可以通过拖拽照片到相册或者点击回收利用图标来还原照片。

您可以通过点击缩放图标来查看大图。jQuery UI 对话框（dialog）部件用于模态窗口。

```
<!doctype html>
<html lang="en">
<head>
```

```
<meta charset="utf-8">
<title>jQuery UI 放置（Droppable） - 简单的照片管理器</title>
<link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
<script src="//code.jquery.com/jquery-1.9.1.js"></script>
<script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
<link rel="stylesheet" href="http://jqueryui.com/resources/demos/
<style>
#gallery { float: left; width: 65%; min-height: 12em; }
.gallery.custom-state-active { background: #eee; }
.gallery li { float: left; width: 96px; padding: 0.4em; margin: 0
.gallery li h5 { margin: 0 0 0.4em; cursor: move; }
.gallery li a { float: right; }
.gallery li a.ui-icon-zoomin { float: left; }
.gallery li img { width: 100%; cursor: move; }

#trash { float: right; width: 32%; min-height: 18em; padding: 1%;
#trash h4 { line-height: 16px; margin: 0 0 0.4em; }
#trash h4 .ui-icon { float: left; }
#trash .gallery h5 { display: none; }
</style>
<script>
$(function() {
  // 这是相册和回收站
  var $gallery = $( "#gallery" ),
    $trash = $( "#trash" );

  // 让相册的条目可拖拽
  $( "li", $gallery ).draggable({
    cancel: "a.ui-icon", // 点击一个图标不会启动拖拽
    revert: "invalid", // 当未被放置时，条目会还原回它的初始位置
    containment: "document",
    helper: "clone",
    cursor: "move"
  });

  // 让回收站可放置，接受相册的条目
  $trash.droppable({
    accept: "#gallery > li",
    activeClass: "ui-state-highlight",
    drop: function( event, ui ) {
      deleteImage( ui.draggable );
    }
  });

  // 让相册可放置，接受回收站的条目
  $gallery.droppable({
    accept: "#trash li",
    activeClass: "custom-state-active",
    drop: function( event, ui ) {
      recycleImage( ui.draggable );
    }
  });
```

```
    // 图像删除功能
    var recycle_icon = "<a href='link/to/recycle/script/when/we/hav
    function deleteImage( $item ) {
      $item.fadeOut(function() {
        var $list = $( "ul", $trash ).length ?
          $( "ul", $trash ) :
          $( "<ul class='gallery ui-helper-reset'/>" ).appendTo( $t

        $item.find( "a.ui-icon-trash" ).remove();
        $item.append( recycle_icon ).appendTo( $list ).fadeIn(funct
          $item
            .animate({ width: "48px" })
            .find( "img" )
              .animate({ height: "36px" });
        });
      });
    }

    // 图像还原功能
    var trash_icon = "<a href='link/to/trash/script/when/we/have/js
    function recycleImage( $item ) {
      $item.fadeOut(function() {
        $item
          .find( "a.ui-icon-refresh" )
            .remove()
          .end()
          .css( "width", "96px")
          .append( trash_icon )
          .find( "img" )
            .css( "height", "72px" )
          .end()
          .appendTo( $gallery )
          .fadeIn();
      });
    }

    // 图像预览功能，演示 ui.dialog 作为模态窗口使用
    function viewLargerImage( $link ) {
      var src = $link.attr( "href" ),
        title = $link.siblings( "img" ).attr( "alt" ),
        $modal = $( "img[src$='" + src + "']" );

      if ( $modal.length ) {
        $modal.dialog( "open" );
      } else {
        var img = $( "<img alt='" + title + "' width='384' height=
          .attr( "src", src ).appendTo( "body" );
        setTimeout(function() {
          img.dialog({
            title: title,
            width: 400,
            modal: true
          });
```

```
        }, 1 );
      }
    }

    // 通过事件代理解决图标行为
    $( "ul.gallery > li" ).click(function( event ) {
      var $item = $( this ),
        $target = $( event.target );

      if ( $target.is( "a.ui-icon-trash" ) ) {
        deleteImage( $item );
      } else if ( $target.is( "a.ui-icon-zoomin" ) ) {
        viewLargerImage( $target );
      } else if ( $target.is( "a.ui-icon-refresh" ) ) {
        recycleImage( $item );
      }

      return false;
    });
  });
  </script>
</head>
<body>

<div class="ui-widget ui-helper-clearfix">

<ul id="gallery" class="gallery ui-helper-reset ui-helper-clearfix'
  <li class="ui-widget-content ui-corner-tr">
    <h5 class="ui-widget-header">High Tatras</h5>
    <img src="img/high_tatras_min.jpg" alt="High Tatras 的最高峰" wi
    <a href="img/high_tatras.jpg" title="查看大图" class="ui-icon ui
    <a href="link/to/trash/script/when/we/have/js/off" title="删除图
  </li>
  <li class="ui-widget-content ui-corner-tr">
    <h5 class="ui-widget-header">High Tatras 2</h5>
    <img src="img/high_tatras2_min.jpg" alt="绿山湖畔的小屋" width="9
    <a href="img/high_tatras2.jpg" title="查看大图" class="ui-icon u
    <a href="link/to/trash/script/when/we/have/js/off" title="删除图
  </li>
  <li class="ui-widget-content ui-corner-tr">
    <h5 class="ui-widget-header">High Tatras 3</h5>
    <img src="img/high_tatras3_min.jpg" alt="计划登高" width="96" he
    <a href="img/high_tatras3.jpg" title="查看大图" class="ui-icon u
    <a href="link/to/trash/script/when/we/have/js/off" title="删除图
  </li>
  <li class="ui-widget-content ui-corner-tr">
    <h5 class="ui-widget-header">High Tatras 4</h5>
    <img src="img/high_tatras4_min.jpg" alt="在 Kozi kopka 的顶部" w
    <a href="img/high_tatras4.jpg" title="查看大图" class="ui-icon u
    <a href="link/to/trash/script/when/we/have/js/off" title="删除图
  </li>
</ul>
```

```
<div id="trash" class="ui-widget-content ui-state-default">
  <h4 class="ui-widget-header"><span class="ui-icon ui-icon-trash">
</div>

</div>

</body>
</html>
```

视觉反馈

当悬停在 droppable 上时，或者当 droppable 被激活（即一个可接受的 draggable 被放置在 droppable 上）时，改变 droppable 的外观。使用 `hoverClass` 或 `activeClass` 选项来分别指定 class。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 放置（Droppable） - 视觉反馈</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #draggable, #draggable2 { width: 90px; height: 90px; padding: 0.5
  #droppable, #droppable2 { width: 120px; height: 120px; padding: 0
  h3 { clear: left; }
  </style>
  <script>
  $(function() {
    $( "#draggable" ).draggable();
    $( "#droppable" ).droppable({
      hoverClass: "ui-state-hover",
      drop: function( event, ui ) {
        $( this )
          .addClass( "ui-state-highlight" )
          .find( "p" )
            .html( "Dropped!" );
      }
    });

    $( "#draggable2" ).draggable();
    $( "#droppable2" ).droppable({
      accept: "#draggable2",
      activeClass: "ui-state-default",
      drop: function( event, ui ) {
        $( this )
          .addClass( "ui-state-highlight" )
```

```
          .find( "p" )
            .html( "Dropped!" );
      }
    });
  });
  </script>
</head>
<body>

<h3>当悬停在 droppable 上时的反馈：</h3>

<div id="draggable" class="ui-widget-content">
  <p>请拖拽我到目标</p>
</div>

<div id="droppable" class="ui-widget-header">
  <p>请放置在这里</p>
</div>

<h3>当激活 draggable 时的反馈：</h3>

<div id="draggable2" class="ui-widget-content">
  <p>请拖拽我到目标</p>
</div>

<div id="droppable2" class="ui-widget-header">
  <p>请放置在这里</p>
</div>

</body>
</html>
```

jQuery UI 实例 - 缩放（Resizable）

使用鼠标改变元素的尺寸。

如需了解更多有关 resizable 交互的细节，请查看 API 文档 可调整尺寸小部件（Resizable Widget）。

默认功能

在任意的 DOM 元素上启用 resizable 功能。通过鼠标拖拽右边或底边的边框到所需的宽度或高度。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 缩放（Resizable） - 默认功能</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #resizable { width: 150px; height: 150px; padding: 0.5em; }
  #resizable h3 { text-align: center; margin: 0; }
  </style>
  <script>
  $(function() {
    $( "#resizable" ).resizable();
  });
  </script>
</head>
<body>

<div id="resizable" class="ui-widget-content">
  <h3 class="ui-widget-header">缩放（Resizable）</h3>
</div>

</body>
</html>
```

动画

使用 `animate` 选项（布尔值）使缩放行为动画化。当该选项设置为 true 时，拖拽轮廓到所需的位置，元素会在拖拽停止时以动画形式调整到该尺寸。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 缩放（Resizable） - 动画</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #resizable { width: 150px; height: 150px; padding: 0.5em; }
  #resizable h3 { text-align: center; margin: 0; }
  .ui-resizable-helper { border: 1px dotted gray; }
  </style>
  <script>
  $(function() {
    $( "#resizable" ).resizable({
      animate: true
    });
  });
  </script>
</head>
<body>

<div id="resizable" class="ui-widget-content">
  <h3 class="ui-widget-header">动画</h3>
</div>

</body>
</html>
```

限制缩放区域

定义缩放区域的边界。使用 `containment` 选项来指定一个父级的 DOM 元素或一个 jQuery 选择器，比如 'document.'。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 缩放（Resizable） - 限制缩放区域</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #container { width: 300px; height: 300px; }
  #container h3 { text-align: center; margin: 0; margin-bottom: 10p
  #resizable { background-position: top left; width: 150px; height
  #resizable, #container { padding: 0.5em; }
  </style>
  <script>
  $(function() {
    $( "#resizable" ).resizable({
      containment: "#container"
    });
  });
  </script>
</head>
<body>

<div id="container" class="ui-widget-content">
  <h3 class="ui-widget-header">限制</h3>
  <div id="resizable" class="ui-state-active">
    <h3 class="ui-widget-header">缩放（Resizable）</h3>
  </div>
</div>

</body>
</html>
```

延迟开始

通过 `delay` 选项设置延迟开始缩放的毫秒数。通过 `distance` 选项设置光标被按下且拖拽指定像素后才允许缩放。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 缩放（Resizable） - 延迟开始</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #resizable, #resizable2 { width: 150px; height: 150px; padding: 0
  #resizable h3, #resizable2 h3 { text-align: center; margin: 0; }
  </style>
  <script>
  $(function() {
    $( "#resizable" ).resizable({
      delay: 1000
    });

    $( "#resizable2" ).resizable({
      distance: 40
    });
  });
  </script>
</head>
<body>

<h3 class="docs">时间延迟 (ms):</h3>
<div id="resizable" class="ui-widget-content">
  <h3 class="ui-widget-header">时间</h3>
</div>

<h3 class="docs">距离延迟 (px):</h3>
<div id="resizable2" class="ui-widget-content">
  <h3 class="ui-widget-header">距离</h3>
</div>

</body>
</html>
```

助手

通过设置 `helper` 选项为一个 CSS class，当缩放时只显示元素的轮廓。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 缩放（Resizable） - 助手</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #resizable { width: 150px; height: 150px; padding: 0.5em; }
  #resizable h3 { text-align: center; margin: 0; }
  .ui-resizable-helper { border: 2px dotted #00F; }
  </style>
  <script>
  $(function() {
    $( "#resizable" ).resizable({
      helper: "ui-resizable-helper"
    });
  });
  </script>
</head>
<body>

<div id="resizable" class="ui-widget-content">
  <h3 class="ui-widget-header">助手</h3>
</div>

</body>
</html>
```

最大/最小尺寸

使用 `maxHeight`、`maxWidth`、`minHeight` 和 `minWidth` 选项限制 resizable 元素的最大或最小高度或宽度。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 缩放（Resizable） - 最大/最小尺寸</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #resizable { width: 200px; height: 150px; padding: 5px; }
  #resizable h3 { text-align: center; margin: 0; }
  </style>
  <script>
  $(function() {
    $( "#resizable" ).resizable({
      maxHeight: 250,
      maxWidth: 350,
      minHeight: 150,
      minWidth: 200
    });
  });
  </script>
</head>
<body>

<div id="resizable" class="ui-widget-content">
  <h3 class="ui-widget-header">放大/缩小</h3>
</div>

</body>
</html>
```

保持纵横比

保持现有的纵横比或设置一个新的纵横比来限制缩放比例。设置 `aspectRatio` 选项为 true，且可选地传递一个新的比率（比如，4/3）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 缩放（Resizable） - 保持纵横比</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #resizable { width: 160px; height: 90px; padding: 0.5em; }
  #resizable h3 { text-align: center; margin: 0; }
  </style>
  <script>
  $(function() {
    $( "#resizable" ).resizable({
      aspectRatio: 16 / 9
    });
  });
  </script>
</head>
<body>

<div id="resizable" class="ui-widget-content">
  <h3 class="ui-widget-header">保持纵横比</h3>
</div>

</body>
</html>
```

对齐到网格

对齐 resizable 元素到网格。通过 `grid` 选项设置网格单元的尺寸（以像素为单位的高度和宽度）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 缩放（Resizable） - 对齐到网格</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #resizable { width: 150px; height: 150px; padding: 0.5em; }
  #resizable h3 { text-align: center; margin: 0; }
  </style>
  <script>
  $(function() {
    $( "#resizable" ).resizable({
      grid: 50
    });
  });
  </script>
</head>
<body>

<div id="resizable" class="ui-widget-content">
  <h3 class="ui-widget-header">网格</h3>
</div>

</body>
</html>
```

同步缩放

通过点击并拖拽一个元素的边来同时调整多个元素的尺寸。给 `alsoResize` 选项传递一个共享的选择器。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 缩放（Resizable） - 同步缩放</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #resizable { background-position: top left; }
  #resizable, #also { width: 150px; height: 120px; padding: 0.5em;
  #resizable h3, #also h3 { text-align: center; margin: 0; }
  #also { margin-top: 1em; }
  </style>
  <script>
  $(function() {
    $( "#resizable" ).resizable({
      alsoResize: "#also"
    });
    $( "#also" ).resizable();
  });
  </script>
</head>
<body>

<div id="resizable" class="ui-widget-header">
  <h3 class="ui-state-active">缩放</h3>
</div>

<div id="also" class="ui-widget-content">
  <h3 class="ui-widget-header">同步缩放</h3>
</div>

</body>
</html>
```

文本框

可缩放的文本框。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 缩放（Resizable） - 文本框</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  .ui-resizable-se {
    bottom: 17px;
  }
  </style>
  <script>
  $(function() {
    $( "#resizable" ).resizable({
      handles: "se"
    });
  });
  </script>
</head>
<body>

<textarea id="resizable" rows="5" cols="20"></textarea>

</body>
</html>
```

视觉反馈

通过设置 `ghost` 选项为 true，可在缩放期间显示一个半透明的元素，而不是显示一个实际的元素。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 缩放（Resizable） - 视觉反馈</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #resizable { width: 150px; height: 150px; padding: 0.5em; }
  #resizable h3 { text-align: center; margin: 0; }
  .ui-resizable-ghost { border: 1px dotted gray; }
  </style>
  <script>
  $(function() {
    $( "#resizable" ).resizable({
      ghost: true
    });
  });
  </script>
</head>
<body>

<div id="resizable" class="ui-widget-content">
  <h3 class="ui-widget-header">Ghost</h3>
</div>

</body>
</html>
```

jQuery UI 实例 - 选择（Selectable）

使用鼠标选择单个元素或一组元素。

如需了解更多有关 selectable 交互的细节，请查看 API 文档 可选择小部件（Selectable Widget）。

默认功能

在某个 DOM 元素上或者一组元素上启用 selectable 功能。通过鼠标拖拽选择条目。按住 Ctrl 键，选择多个不相邻的条目。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 选择（Selectable） - 默认功能</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/

  <style>
  #feedback { font-size: 1.4em; }
  #selectable .ui-selecting { background: #FECA40; }
  #selectable .ui-selected { background: #F39814; color: white; }
  #selectable { list-style-type: none; margin: 0; padding: 0; widtl
  #selectable li { margin: 3px; padding: 0.4em; font-size: 1.4em; l
  </style>
  <script>
  $(function() {
    $( "#selectable" ).selectable();
  });
  </script>
</head>
<body>

<ol id="selectable">
  <li class="ui-widget-content">Item 1</li>
  <li class="ui-widget-content">Item 2</li>
  <li class="ui-widget-content">Item 3</li>
  <li class="ui-widget-content">Item 4</li>
  <li class="ui-widget-content">Item 5</li>
  <li class="ui-widget-content">Item 6</li>
  <li class="ui-widget-content">Item 7</li>
</ol>

</body>
</html>
```

显示为网格

让 selectable 条目显示为网格，使用 CSS 使得它们带有相同的尺寸且浮动显示。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 选择（Selectable） - 显示为网格</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/

  <style>
  #feedback { font-size: 1.4em; }
  #selectable .ui-selecting { background: #FECA40; }
  #selectable .ui-selected { background: #F39814; color: white; }
  #selectable { list-style-type: none; margin: 0; padding: 0; width
  #selectable li { margin: 3px; padding: 1px; float: left; width: 1
  </style>
  <script>
  $(function() {
    $( "#selectable" ).selectable();
  });
  </script>
</head>
<body>

<ol id="selectable">
  <li class="ui-state-default">1</li>
  <li class="ui-state-default">2</li>
  <li class="ui-state-default">3</li>
  <li class="ui-state-default">4</li>
  <li class="ui-state-default">5</li>
  <li class="ui-state-default">6</li>
  <li class="ui-state-default">7</li>
  <li class="ui-state-default">8</li>
  <li class="ui-state-default">9</li>
  <li class="ui-state-default">10</li>
  <li class="ui-state-default">11</li>
  <li class="ui-state-default">12</li>
</ol>

</body>
</html>
```

序列化

写一个函数，在 `stop` 事件发生时触发，来收集被选中条目的索引值。呈现这些值作为反馈，或者以数据字符串形式进行传递。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 选择（Selectable） - 序列化</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/

  <style>
  #feedback { font-size: 1.4em; }
  #selectable .ui-selecting { background: #FECA40; }
  #selectable .ui-selected { background: #F39814; color: white; }
  #selectable { list-style-type: none; margin: 0; padding: 0; widtl
  #selectable li { margin: 3px; padding: 0.4em; font-size: 1.4em; l
  </style>
  <script>
  $(function() {
    $( "#selectable" ).selectable({
      stop: function() {
        var result = $( "#select-result" ).empty();
        $( ".ui-selected", this ).each(function() {
          var index = $( "#selectable li" ).index( this );
          result.append( " #" + ( index + 1 ) );
        });
      }
    });
  });
  </script>
</head>
<body>

<p id="feedback">
<span>您已经选择了：</span> <span id="select-result">无</span>。
</p>

<ol id="selectable">
  <li class="ui-widget-content">Item 1</li>
  <li class="ui-widget-content">Item 2</li>
  <li class="ui-widget-content">Item 3</li>
  <li class="ui-widget-content">Item 4</li>
  <li class="ui-widget-content">Item 5</li>
  <li class="ui-widget-content">Item 6</li>
</ol>

</body>
</html>
```

jQuery UI 实例 - 排序（Sortable）

使用鼠标调整列表中或者网格中元素的排序。

如需了解更多有关 sortable 交互的细节，请查看 API 文档 可排序小部件（Sortable Widget）。

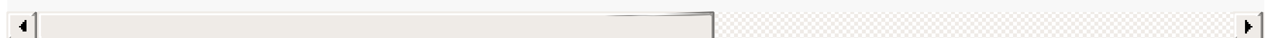
默认功能

在任意的 DOM 元素上启用 sortable 功能。通过鼠标点击并拖拽元素到列表中的一个新的位置，其它条目会自动调整。默认情况下，sortable 各个条目共享 `draggable` 属性。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 排序（Sortable） - 默认功能</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #sortable { list-style-type: none; margin: 0; padding: 0; width:
  #sortable li { margin: 0 3px 3px 3px; padding: 0.4em; padding-lef
  #sortable li span { position: absolute; margin-left: -1.3em; }
  </style>
  <script>
  $(function() {
    $( "#sortable" ).sortable();
    $( "#sortable" ).disableSelection();
  });
  </script>
</head>
<body>

<ul id="sortable">
  <li class="ui-state-default"><span class="ui-icon ui-icon-arrowth
  <li class="ui-state-default"><span class="ui-icon ui-icon-arrowth
  <li class="ui-state-default"><span class="ui-icon ui-icon-arrowth
  <li class="ui-state-default"><span class="ui-icon ui-icon-arrowth
  <li class="ui-state-default"><span class="ui-icon ui-icon-arrowth
  <li class="ui-state-default"><span class="ui-icon ui-icon-arrowth
  <li class="ui-state-default"><span class="ui-icon ui-icon-arrowth
</ul>

</body>
</html>
```



连接列表

通过向 `connectWith` 选项传递一个选择器，来把一个列表中的元素排序到另一个列表中，反之亦然。最简单的办法是将带有某个 CSS class 的所有相关的列表分组，然后传递该 class 到 sortable 函数（比如， `connectWith: '.myclass'` ）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 排序（Sortable） - 连接列表</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #sortable1, #sortable2 { list-style-type: none; margin: 0; paddin
  #sortable1 li, #sortable2 li { margin: 0 5px 5px 5px; padding: 5p
  </style>
  <script>
  $(function() {
    $( "#sortable1, #sortable2" ).sortable({
      connectWith: ".connectedSortable"
    }).disableSelection();
  });
  </script>
</head>
<body>

<ul id="sortable1" class="connectedSortable">
  <li class="ui-state-default">Item 1</li>
  <li class="ui-state-default">Item 2</li>
  <li class="ui-state-default">Item 3</li>
  <li class="ui-state-default">Item 4</li>
  <li class="ui-state-default">Item 5</li>
</ul>

<ul id="sortable2" class="connectedSortable">
  <li class="ui-state-highlight">Item 1</li>
  <li class="ui-state-highlight">Item 2</li>
  <li class="ui-state-highlight">Item 3</li>
  <li class="ui-state-highlight">Item 4</li>
  <li class="ui-state-highlight">Item 5</li>
</ul>

</body>
</html>
```

标签页连接列表

通过放置列表项到顶部适当的标签页中，来把一个列表中的元素排序到另一个列表中，反之亦然。

```
<!doctype html>
```

```
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 排序（Sortable） - 标签页连接列表</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #sortable1 li, #sortable2 li { margin: 0 5px 5px 5px; padding: 5p
  </style>
  <script>
  $(function() {
    $( "#sortable1, #sortable2" ).sortable().disableSelection();

    var $tabs = $( "#tabs" ).tabs();

    var $tab_items = $( "ul:first li", $tabs ).droppable({
      accept: ".connectedSortable li",
      hoverClass: "ui-state-hover",
      drop: function( event, ui ) {
        var $item = $( this );
        var $list = $( $item.find( "a" ).attr( "href" ) )
          .find( ".connectedSortable" );

        ui.draggable.hide( "slow", function() {
          $tabs.tabs( "option", "active", $tab_items.index( $item )
          $( this ).appendTo( $list ).show( "slow" );
        });
      }
    });
  });
  </script>
</head>
<body>

<div id="tabs">
  <ul>
    <li><a href="#tabs-1">Nunc tincidunt</a></li>
    <li><a href="#tabs-2">Proin dolor</a></li>
  </ul>
  <div id="tabs-1">
    <ul id="sortable1" class="connectedSortable ui-helper-reset">
      <li class="ui-state-default">Item 1</li>
      <li class="ui-state-default">Item 2</li>
      <li class="ui-state-default">Item 3</li>
      <li class="ui-state-default">Item 4</li>
      <li class="ui-state-default">Item 5</li>
    </ul>
  </div>
  <div id="tabs-2">
    <ul id="sortable2" class="connectedSortable ui-helper-reset">
      <li class="ui-state-highlight">Item 1</li>
```

```
      <li class="ui-state-highlight">Item 2</li>
      <li class="ui-state-highlight">Item 3</li>
      <li class="ui-state-highlight">Item 4</li>
      <li class="ui-state-highlight">Item 5</li>
    </ul>
  </div>
</div>

</body>
</html>
```

延迟开始

通过时间延迟和距离延迟来防止意外的排序。通过 `delay` 选项设置开始排序之前必须拖拽的毫秒数。通过 `distance` 选项设置开始排序之前必须拖拽的像素数。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 排序（Sortable） - 延迟开始</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #sortable1, #sortable2 { list-style-type: none; margin: 0; paddir
  #sortable1 li, #sortable2 li { margin: 0 5px 5px 5px; padding: 5p
  </style>
  <script>
  $(function() {
    $( "#sortable1" ).sortable({
      delay: 300
    });

    $( "#sortable2" ).sortable({
      distance: 15
    });

    $( "li" ).disableSelection();
  });
  </script>
</head>
<body>

<h3 class="docs">时间延迟 300ms：</h3>

<ul id="sortable1">
  <li class="ui-state-default">Item 1</li>
  <li class="ui-state-default">Item 2</li>
  <li class="ui-state-default">Item 3</li>
  <li class="ui-state-default">Item 4</li>
</ul>

<h3 class="docs">距离延迟 15px：</h3>

<ul id="sortable2">
  <li class="ui-state-default">Item 1</li>
  <li class="ui-state-default">Item 2</li>
  <li class="ui-state-default">Item 3</li>
  <li class="ui-state-default">Item 4</li>
</ul>

</body>
</html>
```

显示为网格

让 sortable 条目显示为网格，使用 CSS 使得它们带有相同的尺寸且浮动显示。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 排序（Sortable） - 显示为网格</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #sortable { list-style-type: none; margin: 0; padding: 0; width:
  #sortable li { margin: 3px 3px 3px 0; padding: 1px; float: left;
  </style>
  <script>
  $(function() {
    $( "#sortable" ).sortable();
    $( "#sortable" ).disableSelection();
  });
  </script>
</head>
<body>

<ul id="sortable">
  <li class="ui-state-default">1</li>
  <li class="ui-state-default">2</li>
  <li class="ui-state-default">3</li>
  <li class="ui-state-default">4</li>
  <li class="ui-state-default">5</li>
  <li class="ui-state-default">6</li>
  <li class="ui-state-default">7</li>
  <li class="ui-state-default">8</li>
  <li class="ui-state-default">9</li>
  <li class="ui-state-default">10</li>
  <li class="ui-state-default">11</li>
  <li class="ui-state-default">12</li>
</ul>

</body>
</html>
```

放置占位符

当拖拽一个 sortable 条目到一个新的位置时，其他条目将为该条目腾出空间。向 `placeholder` 选项传递一个 class 来定义可视化的空白的样式。使用布尔值的 `forcePlaceholderSize` 选项来设置占位符的尺寸。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 排序（Sortable） - 放置占位符</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #sortable { list-style-type: none; margin: 0; padding: 0; width:
  #sortable li { margin: 0 5px 5px 5px; padding: 5px; font-size: 1.
  html>body #sortable li { height: 1.5em; line-height: 1.2em; }
  .ui-state-highlight { height: 1.5em; line-height: 1.2em; }
  </style>
  <script>
  $(function() {
    $( "#sortable" ).sortable({
      placeholder: "ui-state-highlight"
    });
    $( "#sortable" ).disableSelection();
  });
  </script>
</head>
<body>

<ul id="sortable">
  <li class="ui-state-default">Item 1</li>
  <li class="ui-state-default">Item 2</li>
  <li class="ui-state-default">Item 3</li>
  <li class="ui-state-default">Item 4</li>
  <li class="ui-state-default">Item 5</li>
  <li class="ui-state-default">Item 6</li>
  <li class="ui-state-default">Item 7</li>
</ul>

</body>
</html>
```

处理空列表

通过把 选项设置为 `false`，来阻止一个列表中的所有条目被放置到一个单独的空列表中。默认情况下，sortable 条目可被放置到空的列表中。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 排序（Sortable） - 处理空列表</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #sortable1, #sortable2, #sortable3 { list-style-type: none; margi
  #sortable1 li, #sortable2 li, #sortable3 li { margin: 5px; paddin
  </style>
  <script>
  $(function() {
    $( "ul.droptrue" ).sortable({
      connectWith: "ul"
    });

    $( "ul.dropfalse" ).sortable({
      connectWith: "ul",
      dropOnEmpty: false
    });

    $( "#sortable1, #sortable2, #sortable3" ).disableSelection();
  });
  </script>
</head>
<body>

<ul id="sortable1" class="droptrue">
  <li class="ui-state-default">可被放置到..</li>
  <li class="ui-state-default">..一个空列表中</li>
  <li class="ui-state-default">Item 3</li>
  <li class="ui-state-default">Item 4</li>
  <li class="ui-state-default">Item 5</li>
</ul>

<ul id="sortable2" class="dropfalse">
  <li class="ui-state-highlight">不可被放置到..</li>
  <li class="ui-state-highlight">..一个空列表中</li>
  <li class="ui-state-highlight">Item 3</li>
  <li class="ui-state-highlight">Item 4</li>
  <li class="ui-state-highlight">Item 5</li>
</ul>

<ul id="sortable3" class="droptrue">
</ul>

<br style="clear:both">

</body>
```

```
</html>
```

包含/排除条目

指定通过向 `items` 选项传递一个 jQuery 选择器哪些项目可排序。该选项之外的项目则是不可排序的，同时它们也不是 sortable 条目的有效的目标。

如果只想防止特定的条目排序，则向 `cancel` 选项传递一个 jQuery 选择器。已取消的条目依然是其他条目的有效的排序目标。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 排序（Sortable） - 包含/排除条目</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #sortable1, #sortable2 { list-style-type: none; margin: 0; paddin
  #sortable1 li, #sortable2 li { margin: 0 5px 5px 5px; padding: 3p
  </style>
  <script>
  $(function() {
    $( "#sortable1" ).sortable({
      items: "li:not(.ui-state-disabled)"
    });

    $( "#sortable2" ).sortable({
      cancel: ".ui-state-disabled"
    });

    $( "#sortable1 li, #sortable2 li" ).disableSelection();
  });
  </script>
</head>
<body>

<h3 class="docs">指定哪个条目是 sortable：</h3>

<ul id="sortable1">
  <li class="ui-state-default">Item 1</li>
  <li class="ui-state-default ui-state-disabled">（我不是 sortable 或
  <li class="ui-state-default ui-state-disabled">（我不是 sortable 或
  <li class="ui-state-default">Item 4</li>
</ul>

<h3 class="docs">取消排序（但作为放置目标）：</h3>

<ul id="sortable2">
  <li class="ui-state-default">Item 1</li>
  <li class="ui-state-default ui-state-disabled">（我不是 sortable）
  <li class="ui-state-default ui-state-disabled">（我不是 sortable）
  <li class="ui-state-default">Item 4</li>
</ul>

</body>
</html>
```

门户组件（Portlets）

启用门户组件（样式化的 div）作为 sortable，并使用 `connectWith` 选项来允许在列之间进行排序。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 排序（Sortable） - 门户组件（Portlets）</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  body {
    min-width: 520px;
  }
  .column {
    width: 170px;
    float: left;
    padding-bottom: 100px;
  }
  .portlet {
    margin: 0 1em 1em 0;
    padding: 0.3em;
  }
  .portlet-header {
    padding: 0.2em 0.3em;
    margin-bottom: 0.5em;
    position: relative;
  }
  .portlet-toggle {
    position: absolute;
    top: 50%;
    right: 0;
    margin-top: -8px;
  }
  .portlet-content {
    padding: 0.4em;
  }
  .portlet-placeholder {
    border: 1px dotted black;
    margin: 0 1em 1em 0;
    height: 50px;
  }
  </style>
  <script>
  $(function() {
    $( ".column" ).sortable({
      connectWith: ".column",
```

```
      handle: ".portlet-header",
      cancel: ".portlet-toggle",
      placeholder: "portlet-placeholder ui-corner-all"
    });

    $( ".portlet" )
      .addClass( "ui-widget ui-widget-content ui-helper-clearfix ui
      .find( ".portlet-header" )
        .addClass( "ui-widget-header ui-corner-all" )
        .prepend( "<span class='ui-icon ui-icon-minusthick portlet-

    $( ".portlet-toggle" ).click(function() {
      var icon = $( this );
      icon.toggleClass( "ui-icon-minusthick ui-icon-plusthick" );
      icon.closest( ".portlet" ).find( ".portlet-content" ).toggle(
    });
  });
  </script>
</head>
<body>

<div class="column">

  <div class="portlet">
    <div class="portlet-header">订阅</div>
    <div class="portlet-content">Lorem ipsum dolor sit amet, consec
  </div>

  <div class="portlet">
    <div class="portlet-header">新闻</div>
    <div class="portlet-content">Lorem ipsum dolor sit amet, consec
  </div>

</div>

<div class="column">

  <div class="portlet">
    <div class="portlet-header">购物</div>
    <div class="portlet-content">Lorem ipsum dolor sit amet, consec
  </div>

</div>

<div class="column">

  <div class="portlet">
    <div class="portlet-header">链接</div>
    <div class="portlet-content">Lorem ipsum dolor sit amet, consec
  </div>

  <div class="portlet">
    <div class="portlet-header">图像</div>
```

```
    <div class="portlet-content">Lorem ipsum dolor sit amet, consec
  </div>

</div>

</body>
</html>
```

jQuery UI 实例 - 折叠面板（Accordion）

在一个有限的空间内显示用于呈现信息的可折叠的内容面板。

如需了解更多有关 accordion 部件的细节，请查看 API 文档 折叠面板部件（Accordion Widget）。

默认功能

点击头部展开/折叠被分为各个逻辑部分的内容，就像标签页（tabs）一样。您可以选择性地设置当鼠标悬停时是否切换各部分的打开/关闭状态。

基本的 HTML 标记是一系列的标题（H3 标签）和内容 div，因此内容不用通过 JavaScript 即可用。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 折叠面板（Accordion） - 默认功能</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#accordion" ).accordion();
  });
  </script>
</head>
<body>

<div id="accordion">
  <h3>部分 1</h3>
  <div>
    <p>
    Mauris mauris ante, blandit et, ultrices a, suscipit eget, quar
    ut neque. Vivamus nisi metus, molestie vel, gravida in, condime
    amet, nunc. Nam a nibh. Donec suscipit eros. Nam mi. Proin vive
    odio. Curabitur malesuada. Vestibulum a velit eu ante scelerisc
    </p>
  </div>
  <h3>部分 2</h3>
  <div>
    <p>
    Sed non urna. Donec et ante. Phasellus eu ligula. Vestibulum si
    purus. Vivamus hendrerit, dolor at aliquet laoreet, mauris turp
    velit, faucibus interdum tellus libero ac justo. Vivamus non qu
```

```
    suscipit faucibus urna.
    </p>
  </div>
  <h3>部分 3</h3>
  <div>
    <p>
    Nam enim risus, molestie et, porta ac, aliquam ac, risus. Quisc
    Phasellus pellentesque purus in massa. Aenean in pede. Phasellu
    ac tellus pellentesque semper. Sed ac felis. Sed commodo, magna
    lacinia ornare, quam ante aliquam nisi, eu iaculis leo purus ve
    </p>
    <ul>
      <li>List item one</li>
      <li>List item two</li>
      <li>List item three</li>
    </ul>
  </div>
  <h3>部分 4</h3>
  <div>
    <p>
    Cras dictum. Pellentesque habitant morbi tristique senectus et
    et malesuada fames ac turpis egestas. Vestibulum ante ipsum pri
    faucibus orci luctus et ultrices posuere cubilia Curae; Aenean
    mauris vel est.
    </p>
    <p>
    Suspendisse eu nisl. Nullam ut libero. Integer dignissim consec
    Class aptent taciti sociosqu ad litora torquent per conubia nos
    inceptos himenaeos.
    </p>
  </div>
</div>

</body>
</html>
```

折叠内容

默认情况下，折叠面板总是保持一个部分是打开的。为了让所有部分都是折叠的，可设置 `collapsible` 选项为 true。点击当前打开的部分，来折叠它的内容面板。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 折叠面板（Accordion） - 折叠内容</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#accordion" ).accordion({
      collapsible: true
    });
  });
  </script>
</head>
<body>

<div id="accordion">
  <h3>部分 1</h3>
  <div>
    <p>Mauris mauris ante, blandit et, ultrices a, suscipit eget, q
  </div>
  <h3>部分 2</h3>
  <div>
    <p>Sed non urna. Donec et ante. Phasellus eu ligula. Vestibulum
  </div>
  <h3>部分 3</h3>
  <div>
    <p>Nam enim risus, molestie et, porta ac, aliquam ac, risus. Qu
    <ul>
      <li>List item one</li>
      <li>List item two</li>
      <li>List item three</li>
    </ul>
  </div>
  <h3>部分 4</h3>
  <div>
    <p>Cras dictum. Pellentesque habitant morbi tristique senectus
  </div>
</div>

</body>
</html>
```

自定义图标

通过 `icons` 选项自定义标题图标，`icons` 选项接受标题默认的和激活的（打开的）状态的 class。使用 UI CSS 框架中的任意 class，或者使用背景图像创建自定义的 class。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 折叠面板（Accordion） - 自定义图标</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    var icons = {
      header: "ui-icon-circle-arrow-e",
      activeHeader: "ui-icon-circle-arrow-s"
    };
    $( "#accordion" ).accordion({
      icons: icons
    });
    $( "#toggle" ).button().click(function() {
      if ( $( "#accordion" ).accordion( "option", "icons" ) ) {
        $( "#accordion" ).accordion( "option", "icons", null );
      } else {
        $( "#accordion" ).accordion( "option", "icons", icons );
      }
    });
  });
  </script>
</head>
<body>

<div id="accordion">
  <h3>部分 1</h3>
  <div>
    <p>Mauris mauris ante, blandit et, ultrices a, suscipit eget, q
  </div>
  <h3>部分 2</h3>
  <div>
    <p>Sed non urna. Donec et ante. Phasellus eu ligula. Vestibulum
  </div>
  <h3>部分 3</h3>
  <div>
    <p>Nam enim risus, molestie et, porta ac, aliquam ac, risus. Qu
    <ul>
      <li>List item one</li>
      <li>List item two</li>
      <li>List item three</li>
    </ul>
  </div>
```

```
  <h3>部分 4</h3>
  <div>
    <p>Cras dictum. Pellentesque habitant morbi tristique senectus
  </div>
</div>

<button id="toggle">切换图标</button>

</body>
</html>
```

填充空间

由于折叠面板是由块级元素组成的，默认情况下它的宽度会填充可用的水平空间。为了填充由容器分配的垂直空间，设置 `heightStyle` 选项为 `"fill"`，脚本会自动设置折叠面板的尺寸为父容器的高度。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 折叠面板（Accordion） - 填充空间</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #accordion-resizer {
    padding: 10px;
    width: 350px;
    height: 220px;
  }
  </style>
  <script>
  $(function() {
    $( "#accordion" ).accordion({
      heightStyle: "fill"
    });
  });
  $(function() {
    $( "#accordion-resizer" ).resizable({
      minHeight: 140,
      minWidth: 200,
      resize: function() {
        $( "#accordion" ).accordion( "refresh" );
      }
    });
  });
  </script>
```

```
</head>
<body>

<h3 class="docs">重新调整外部容器：</h3>

<div id="accordion-resizer" class="ui-widget-content">
  <div id="accordion">
    <h3>部分 1</h3>
    <div>
      <p>Mauris mauris ante, blandit et, ultrices a, suscipit eget,
    </div>
    <h3>部分 2</h3>
    <div>
      <p>Sed non urna. Donec et ante. Phasellus eu ligula. Vestibul
    </div>
    <h3>部分 3</h3>
    <div>
      <p>Nam enim risus, molestie et, porta ac, aliquam ac, risus.
      <ul>
        <li>List item one</li>
        <li>List item two</li>
        <li>List item three</li>
      </ul>
    </div>
    <h3>部分 4</h3>
    <div>
      <p>Cras dictum. Pellentesque habitant morbi tristique senectu
    </div>
  </div>
</div>

</body>
</html>
```

非自动高度

设置 `heightStyle: "content"` ，让折叠面板保持它们初始的高度。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 折叠面板（Accordion） - 非自动高度</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#accordion" ).accordion({
      heightStyle: "content"
    });
  });
  </script>
</head>
<body>

<div id="accordion">
  <h3>部分 1</h3>
  <div>
    <p>Mauris mauris ante, blandit et, ultrices a, susceros. Nam mi
  </div>
  <h3>部分 2</h3>
  <div>
    <p>Sed non urna. Donec et ante. Phasellus eu ligula. Vestibulum
  </div>
  <h3>部分 3</h3>
  <div>
    <p>Nam enim risus, molestie et, porta ac, aliquam ac, risus. Qu
    <ul>
      <li>List item</li>
      <li>List item</li>
      <li>List item</li>
      <li>List item</li>
      <li>List item</li>
      <li>List item</li>
      <li>List item</li>
    </ul>
  </div>
</div>

</body>
</html>
```

当悬停时打开

点击头部展开/折叠被分为各个逻辑部分的内容，就像标签页（tabs）一样。您可以选择性地设置当鼠标悬停时是否切换各部分的打开/关闭状态。

基本的 HTML 标记是一系列的标题（H3 标签）和内容 div，因此内容不用通过 JavaScript 即可用。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 折叠面板（Accordion） - 当悬停时打开</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#accordion" ).accordion({
      event: "click hoverintent"
    });
  });

  /*
   * hoverIntent | Copyright 2011 Brian Cherne
   * http://cherne.net/brian/resources/jquery.hoverIntent.html
   * modified by the jQuery UI team
   */
  $.event.special.hoverintent = {
    setup: function() {
      $( this ).bind( "mouseover", jQuery.event.special.hoverintent
    },
    teardown: function() {
      $( this ).unbind( "mouseover", jQuery.event.special.hoverinte
    },
    handler: function( event ) {
      var currentX, currentY, timeout,
        args = arguments,
        target = $( event.target ),
        previousX = event.pageX,
        previousY = event.pageY;

      function track( event ) {
        currentX = event.pageX;
        currentY = event.pageY;
      };

      function clear() {
        target
          .unbind( "mousemove", track )
          .unbind( "mouseout", clear );
        clearTimeout( timeout );
      }
```

```
      function handler() {
        var prop,
          orig = event;

        if ( ( Math.abs( previousX - currentX ) +
            Math.abs( previousY - currentY ) ) < 7 ) {
          clear();

          event = $.Event( "hoverintent" );
          for ( prop in orig ) {
            if ( !( prop in event ) ) {
              event[ prop ] = orig[ prop ];
            }
          }
          // 防止访问原始事件，因为新事件会被异步触发，旧事件不再可用 (#6028
          delete event.originalEvent;

          target.trigger( event );
        } else {
          previousX = currentX;
          previousY = currentY;
          timeout = setTimeout( handler, 100 );
        }
      }

      timeout = setTimeout( handler, 100 );
      target.bind({
        mousemove: track,
        mouseout: clear
      });
    }
  };
  </script>
</head>
<body>

<div id="accordion">
  <h3>部分 1</h3>
  <div>
    <p>
    Mauris mauris ante, blandit et, ultrices a, suscipit eget, quar
    ut neque. Vivamus nisi metus, molestie vel, gravida in, condime
    amet, nunc. Nam a nibh. Donec suscipit eros. Nam mi. Proin vive
    odio. Curabitur malesuada. Vestibulum a velit eu ante sceleriso
    </p>
  </div>
  <h3>部分 2</h3>
  <div>
    <p>
    Sed non urna. Donec et ante. Phasellus eu ligula. Vestibulum si
    purus. Vivamus hendrerit, dolor at aliquet laoreet, mauris turp
    velit, faucibus interdum tellus libero ac justo. Vivamus non qu
```

```
      suscipit faucibus urna.
      </p>
    </div>
    <h3>部分 3</h3>
    <div>
      <p>
      Nam enim risus, molestie et, porta ac, aliquam ac, risus. Quisc
      Phasellus pellentesque purus in massa. Aenean in pede. Phasellu
      ac tellus pellentesque semper. Sed ac felis. Sed commodo, magna
      lacinia ornare, quam ante aliquam nisi, eu iaculis leo purus ve
      </p>
      <ul>
        <li>List item one</li>
        <li>List item two</li>
        <li>List item three</li>
      </ul>
    </div>
    <h3>部分 4</h3>
    <div>
      <p>
      Cras dictum. Pellentesque habitant morbi tristique senectus et
      et malesuada fames ac turpis egestas. Vestibulum ante ipsum pri
      faucibus orci luctus et ultrices posuere cubilia Curae; Aenean
      mauris vel est.
      </p>
      <p>
      Suspendisse eu nisl. Nullam ut libero. Integer dignissim consec
      Class aptent taciti sociosqu ad litora torquent per conubia nos
      inceptos himenaeos.
      </p>
    </div>
  </div>

</body>
</html>
```

排序（Sortable）

拖拽标题来给面板重新排序。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 折叠面板（Accordion） - 排序（Sortable）</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
```

```
  <style>
  /* 当排序时，IE 存在布局问题（查看 #5413） */
  .group { zoom: 1 }
  </style>
  <script>
  $(function() {
    $( "#accordion" )
      .accordion({
        header: "> div > h3"
      })
      .sortable({
        axis: "y",
        handle: "h3",
        stop: function( event, ui ) {
          // 当排序时，IE 不能注册 blur，所以触发 focusout 处理程序来移除
          ui.item.children( "h3" ).triggerHandler( "focusout" );
        }
      });
  });
  </script>
</head>
<body>

<div id="accordion">
  <div class="group">
    <h3>部分 1</h3>
    <div>
      <p>Mauris mauris ante, blandit et, ultrices a, suscipit eget,
    </div>
  </div>
  <div class="group">
    <h3>部分 2</h3>
    <div>
      <p>Sed non urna. Donec et ante. Phasellus eu ligula. Vestibul
    </div>
  </div>
  <div class="group">
    <h3>部分 3</h3>
    <div>
      <p>Nam enim risus, molestie et, porta ac, aliquam ac, risus.
      <ul>
        <li>List item one</li>
        <li>List item two</li>
        <li>List item three</li>
      </ul>
    </div>
  </div>
  <div class="group">
    <h3>部分 4</h3>
    <div>
      <p>Cras dictum. Pellentesque habitant morbi tristique senectu
    </div>
  </div>
```

```
</div>

</body>
</html>
```

jQuery UI 实例 - 自动完成（Autocomplete）

根据用户输入值进行搜索和过滤，让用户快速找到并从预设值列表中选择。

如需了解更多有关 autocomplete 部件的细节，请查看 API 文档 自动完成部件（Autocomplete Widget）。

默认功能

当您在输入域中输入时，自动完成（Autocomplete）部件提供相应的建议。在本实例中，提供了编程语言的建议选项，您可以输入 "ja" 尝试一下，可以得到 Java 或 JavaScript。

数据源是一个简单的 JavaScript 数组，使用 source 选项提供给部件。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 自动完成（Autocomplete） - 默认功能</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    var availableTags = [
      "ActionScript",
      "AppleScript",
      "Asp",
      "BASIC",
      "C",
      "C++",
      "Clojure",
      "COBOL",
      "ColdFusion",
      "Erlang",
      "Fortran",
      "Groovy",
      "Haskell",
      "Java",
      "JavaScript",
      "Lisp",
      "Perl",
      "PHP",
      "Python",
      "Ruby",
      "Scala",
      "Scheme"
    ];
    $( "#tags" ).autocomplete({
      source: availableTags
    });
  });
  </script>
</head>
<body>

<div class="ui-widget">
  <label for="tags">标签：</label>
  <input id="tags">
</div>

</body>
</html>
```

包含重音

autocomplete 域使用自定义的 source 选项来匹配带有重音字符的结果项，即使文本域不包含重音字符也会匹配。但是如果您在文本域中键入了重音字符，则不会显示非重音的结果项。

尝试键入 "Jo"，会看到 "John" 和 "J?rn"，然后 键入 "J?"，只会看到 "J?rn"。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 自动完成（Autocomplete） - 包含重音</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    var names = [ "J?rn Zaefferer", "Scott González", "John Resig"

    var accentMap = {
      "á": "a",
      "?": "o"
    };
    var normalize = function( term ) {
      var ret = "";
      for ( var i = 0; i < term.length; i++ ) {
        ret += accentMap[ term.charAt(i) ] || term.charAt(i);
      }
      return ret;
    };

    $( "#developer" ).autocomplete({
      source: function( request, response ) {
        var matcher = new RegExp( $.ui.autocomplete.escapeRegex( re
        response( $.grep( names, function( value ) {
          value = value.label || value.value || value;
          return matcher.test( value ) || matcher.test( normalize(
        }) );
      }
    });
  });
  </script>
</head>
<body>

<div class="ui-widget">
  <form>
  <label for="developer">开发人员：</label>
  <input id="developer">
  </form>
</div>

</body>
</html>
```

分类

分类的搜索结果。尝试键入 "a" 或 "n"。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 自动完成（Autocomplete） - 分类</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  .ui-autocomplete-category {
    font-weight: bold;
    padding: .2em .4em;
    margin: .8em 0 .2em;
    line-height: 1.5;
  }
  </style>
  <script>
  $.widget( "custom.catcomplete", $.ui.autocomplete, {
    _renderMenu: function( ul, items ) {
      var that = this,
        currentCategory = "";
      $.each( items, function( index, item ) {
        if ( item.category != currentCategory ) {
          ul.append( "<li class='ui-autocomplete-category'>" + iter
          currentCategory = item.category;
        }
        that._renderItemData( ul, item );
      });
    }
  });
  </script>
  <script>
  $(function() {
    var data = [
      { label: "anders", category: "" },
      { label: "andreas", category: "" },
      { label: "antal", category: "" },
      { label: "annhhx10", category: "Products" },
      { label: "annk K12", category: "Products" },
      { label: "annttop C13", category: "Products" },
      { label: "anders andersson", category: "People" },
      { label: "andreas andersson", category: "People" },
      { label: "andreas johnson", category: "People" }
    ];

    $( "#search" ).catcomplete({
```

```
      delay: 0,
      source: data
    });
  });
  </script>
</head>
<body>

<label for="search">搜索：</label>
<input id="search">

</body>
</html>
```

组合框（Combobox）

一个由 Autocomplete 和 Button 创建的自定义部件。您可以键入一些字符，来获得基于您的输入过滤的结果，或者使用按钮从完整列表中选择。

该输入是从一个已有的 select 元素中读取，传递给带有自定义的 source 选项的 Autocomplete。

这是一个不被支持的不完美的部件。这里纯粹是为了演示 autocomplete 定制功能。如需了解更多有关该部件工作原理的细节，请点击这里查看相关的 jQuery 文章。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 自动完成（Autocomplete） - 组合框（Combobox）</titl
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  .custom-combobox {
    position: relative;
    display: inline-block;
  }
  .custom-combobox-toggle {
    position: absolute;
    top: 0;
    bottom: 0;
    margin-left: -1px;
    padding: 0;
    /* 支持： IE7 */
    *height: 1.7em;
    *top: 0.1em;
```

```
  }
  .custom-combobox-input {
    margin: 0;
    padding: 0.3em;
  }
  </style>
  <script>
  (function( $ ) {
    $.widget( "custom.combobox", {
      _create: function() {
        this.wrapper = $( "<span>" )
          .addClass( "custom-combobox" )
          .insertAfter( this.element );

        this.element.hide();
        this._createAutocomplete();
        this._createShowAllButton();
      },

      _createAutocomplete: function() {
        var selected = this.element.children( ":selected" ),
          value = selected.val() ? selected.text() : "";

        this.input = $( "<input>" )
          .appendTo( this.wrapper )
          .val( value )
          .attr( "title", "" )
          .addClass( "custom-combobox-input ui-widget ui-widget-co
          .autocomplete({
            delay: 0,
            minLength: 0,
            source: $.proxy( this, "_source" )
          })
          .tooltip({
            tooltipClass: "ui-state-highlight"
          });

        this._on( this.input, {
          autocompleteselect: function( event, ui ) {
            ui.item.option.selected = true;
            this._trigger( "select", event, {
              item: ui.item.option
            });
          },

          autocompletechange: "_removeIfInvalid"
        });
      },

      _createShowAllButton: function() {
        var input = this.input,
          wasOpen = false;
```

```
        $( "<a>" )
          .attr( "tabIndex", -1 )
          .attr( "title", "Show All Items" )
          .tooltip()
          .appendTo( this.wrapper )
          .button({
            icons: {
              primary: "ui-icon-triangle-1-s"
            },
            text: false
          })
          .removeClass( "ui-corner-all" )
          .addClass( "custom-combobox-toggle ui-corner-right" )
          .mousedown(function() {
            wasOpen = input.autocomplete( "widget" ).is( ":visible'
          })
          .click(function() {
            input.focus();

            // 如果已经可见则关闭
            if ( wasOpen ) {
              return;
            }

            // 传递空字符串作为搜索的值，显示所有的结果
            input.autocomplete( "search", "" );
          });
      },

      _source: function( request, response ) {
        var matcher = new RegExp( $.ui.autocomplete.escapeRegex(rec
        response( this.element.children( "option" ).map(function()
          var text = $( this ).text();
          if ( this.value && ( !request.term || matcher.test(text)
            return {
              label: text,
              value: text,
              option: this
            };
        }) );
      },

      _removeIfInvalid: function( event, ui ) {

        // 选择一项，不执行其他动作
        if ( ui.item ) {
          return;
        }

        // 搜索一个匹配（不区分大小写）
        var value = this.input.val(),
          valueLowerCase = value.toLowerCase(),
          valid = false;
```

```
        this.element.children( "option" ).each(function() {
          if ( $( this ).text().toLowerCase() === valueLowerCase ) {
            this.selected = valid = true;
            return false;
          }
        });

        // 找到一个匹配，不执行其他动作
        if ( valid ) {
          return;
        }

        // 移除无效的值
        this.input
          .val( "" )
          .attr( "title", value + " didn't match any item" )
          .tooltip( "open" );
        this.element.val( "" );
        this._delay(function() {
          this.input.tooltip( "close" ).attr( "title", "" );
        }, 2500 );
        this.input.data( "ui-autocomplete" ).term = "";
      },

      _destroy: function() {
        this.wrapper.remove();
        this.element.show();
      }
    });
  })( jQuery );

  $(function() {
    $( "#combobox" ).combobox();
    $( "#toggle" ).click(function() {
      $( "#combobox" ).toggle();
    });
  });
  </script>
</head>
<body>

<div class="ui-widget">
  <label>您喜欢的编程语言：</label>
  <select id="combobox">
    <option value="">请选择...</option>
    <option value="ActionScript">ActionScript</option>
    <option value="AppleScript">AppleScript</option>
    <option value="Asp">Asp</option>
    <option value="BASIC">BASIC</option>
    <option value="C">C</option>
    <option value="C++">C++</option>
    <option value="Clojure">Clojure</option>
    <option value="COBOL">COBOL</option>
```

```
    <option value="ColdFusion">ColdFusion</option>
    <option value="Erlang">Erlang</option>
    <option value="Fortran">Fortran</option>
    <option value="Groovy">Groovy</option>
    <option value="Haskell">Haskell</option>
    <option value="Java">Java</option>
    <option value="JavaScript">JavaScript</option>
    <option value="Lisp">Lisp</option>
    <option value="Perl">Perl</option>
    <option value="PHP">PHP</option>
    <option value="Python">Python</option>
    <option value="Ruby">Ruby</option>
    <option value="Scala">Scala</option>
    <option value="Scheme">Scheme</option>
  </select>
</div>
<button id="toggle">显示基础的选择框</button>

</body>
</html>
```

自定义数据并显示

您可以使用自定义数据格式，并通过简单地重载默认的聚焦和选择行为来显示数据。

尝试键入 "j"，或者按向下箭头按键，即可得到一个项目列表。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 自动完成（Autocomplete） - 自定义数据并显示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #project-label {
    display: block;
    font-weight: bold;
    margin-bottom: 1em;
  }
  #project-icon {
    float: left;
    height: 32px;
    width: 32px;
  }
  #project-description {
```

```
    margin: 0;
    padding: 0;
  }
  </style>
  <script>
  $(function() {
    var projects = [
      {
        value: "jquery",
        label: "jQuery",
        desc: "the write less, do more, JavaScript library",
        icon: "jquery_32x32.png"
      },
      {
        value: "jquery-ui",
        label: "jQuery UI",
        desc: "the official user interface library for jQuery",
        icon: "jqueryui_32x32.png"
      },
      {
        value: "sizzlejs",
        label: "Sizzle JS",
        desc: "a pure-JavaScript CSS selector engine",
        icon: "sizzlejs_32x32.png"
      }
    ];

    $( "#project" ).autocomplete({
      minLength: 0,
      source: projects,
      focus: function( event, ui ) {
        $( "#project" ).val( ui.item.label );
        return false;
      },
      select: function( event, ui ) {
        $( "#project" ).val( ui.item.label );
        $( "#project-id" ).val( ui.item.value );
        $( "#project-description" ).html( ui.item.desc );
        $( "#project-icon" ).attr( "src", "images/" + ui.item.icon

        return false;
      }
    })
    .data( "ui-autocomplete" )._renderItem = function( ul, item ) 
      return $( "<li>" )
        .append( "<a>" + item.label + "<br>" + item.desc + "</a>" 
        .appendTo( ul );
    };
  });
  </script>
</head>
<body>
```

```
<div id="project-label">选择一个项目（请键入 "j"）：</div>
<img id="project-icon" src="images/transparent_1x1.png" class="ui-s
<input id="project">
<input type="hidden" id="project-id">
<p id="project-description"></p>

</body>
</html>
```

多个值

用法：键入一些字符，比如 "j"，可以看到相关的编程语言结果。选择一个值，然后继续键入字符来添加其他的值。

本实例演示如何使用 source 选项和一些事件来实现在一个单一的文本域输入多个自动完成的值。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 自动完成（Autocomplete） - 多个值</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    var availableTags = [
      "ActionScript",
      "AppleScript",
      "Asp",
      "BASIC",
      "C",
      "C++",
      "Clojure",
      "COBOL",
      "ColdFusion",
      "Erlang",
      "Fortran",
      "Groovy",
      "Haskell",
      "Java",
      "JavaScript",
      "Lisp",
      "Perl",
      "PHP",
      "Python",
      "Ruby",
```

```
      "Scala",
      "Scheme"
    ];
    function split( val ) {
      return val.split( /,\s*/ );
    }
    function extractLast( term ) {
      return split( term ).pop();
    }

    $( "#tags" )
      // 当选择一个条目时不离开文本域
      .bind( "keydown", function( event ) {
        if ( event.keyCode === $.ui.keyCode.TAB &&
            $( this ).data( "ui-autocomplete" ).menu.active ) {
          event.preventDefault();
        }
      })
      .autocomplete({
        minLength: 0,
        source: function( request, response ) {
          // 回到 autocomplete，但是提取最后的条目
          response( $.ui.autocomplete.filter(
            availableTags, extractLast( request.term ) ) );
        },
        focus: function() {
          // 防止在获得焦点时插入值
          return false;
        },
        select: function( event, ui ) {
          var terms = split( this.value );
          // 移除当前输入
          terms.pop();
          // 添加被选项
          terms.push( ui.item.value );
          // 添加占位符，在结尾添加逗号+空格
          terms.push( "" );
          this.value = terms.join( ", " );
          return false;
        }
      });
  });
  </script>
</head>
<body>

<div class="ui-widget">
  <label for="tags">编程语言：</label>
  <input id="tags" size="50">
</div>

</body>
</html>
```



多个值，远程

用法：键入至少两个字符来获取鸟的名称。选择一个值，然后继续键入字符来添加其他的值。

本实例演示如何使用 source 选项和一些事件来实现在一个单一的文本域输入多个自动完成的值。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 自动完成（Autocomplete） - 多个值，远程</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  .ui-autocomplete-loading {
    background: white url('images/ui-anim_basic_16x16.gif') right c
  }
  </style>
  <script>
  $(function() {
    function split( val ) {
      return val.split( /,\s*/ );
    }
    function extractLast( term ) {
      return split( term ).pop();
    }

    $( "#birds" )
      // 当选择一个条目时不离开文本域
      .bind( "keydown", function( event ) {
        if ( event.keyCode === $.ui.keyCode.TAB &&
            $( this ).data( "ui-autocomplete" ).menu.active ) {
          event.preventDefault();
        }
      })
      .autocomplete({
        source: function( request, response ) {
          $.getJSON( "search.php", {
            term: extractLast( request.term )
          }, response );
        },
        search: function() {
          // 自定义最小长度
          var term = extractLast( this.value );
```

```
          if ( term.length < 2 ) {
            return false;
          }
        },
        focus: function() {
          // 防止在获得焦点时插入值
          return false;
        },
        select: function( event, ui ) {
          var terms = split( this.value );
          // 移除当前输入
          terms.pop();
          // 添加被选项
          terms.push( ui.item.value );
          // 添加占位符，在结尾添加逗号+空格
          terms.push( "" );
          this.value = terms.join( ", " );
          return false;
        }
      });
  });
  </script>
</head>
<body>

<div class="ui-widget">
  <label for="birds">鸟：</label>
  <input id="birds" size="50">
</div>

</body>
</html>
```

远程 JSONP 数据源

当您在文本域中键入字符时，Autocomplete 部件给出建议结果。在本实例中，当您在文本域中至少键入两个字符时，将显示相关城市的名称。

在本实例中，数据源是 geonames.org webservice。虽然选择一个元素后文本域中是该城市名称，但是会显示更多的信息以便找到正确的条目。数据也可以回调，显示在下面的结果框中。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 自动完成（Autocomplete） - 远程 JSONP 数据源</title
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
```

```
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  .ui-autocomplete-loading {
    background: white url('images/ui-anim_basic_16x16.gif') right c
  }
  #city { width: 25em; }
  </style>
  <script>
  $(function() {
    function log( message ) {
      $( "<div>" ).text( message ).prependTo( "#log" );
      $( "#log" ).scrollTop( 0 );
    }

    $( "#city" ).autocomplete({
      source: function( request, response ) {
        $.ajax({
          url: "http://ws.geonames.org/searchJSON",
          dataType: "jsonp",
          data: {
            featureClass: "P",
            style: "full",
            maxRows: 12,
            name_startsWith: request.term
          },
          success: function( data ) {
            response( $.map( data.geonames, function( item ) {
              return {
                label: item.name + (item.adminName1 ? ", " + item.a
                value: item.name
              }
            }));
          }
        });
      },
      minLength: 2,
      select: function( event, ui ) {
        log( ui.item ?
          "Selected: " + ui.item.label :
          "Nothing selected, input was " + this.value);
      },
      open: function() {
        $( this ).removeClass( "ui-corner-all" ).addClass( "ui-corr
      },
      close: function() {
        $( this ).removeClass( "ui-corner-top" ).addClass( "ui-corr
      }
    });
  });
  </script>
</head>
<body>
```

```
<div class="ui-widget">
  <label for="city">您的城市：</label>
  <input id="city">
  Powered by <a href="http://geonames.org" target="_blank">geonames
</div>

<div class="ui-widget" style="margin-top:2em; font-family:Arial">
  结果：
  <div id="log" style="height: 200px; width: 300px; overflow: auto;
</div>

</body>
</html>
```

远程数据源

当您在文本域中键入字符时，Autocomplete 部件给出建议结果。在本实例中，当您在文本域中至少键入两个字符时，将显示相关鸟的名称。

在本实例中，数据源是可返回 JSON 数据的服务器端脚本，通过一个简单的 source 选项来指定。另外，minLength 选项设置为 2，避免查询返回太多的结果，select 事件用于显示一些反馈。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 自动完成（Autocomplete） - 远程数据源</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  .ui-autocomplete-loading {
    background: white url('images/ui-anim_basic_16x16.gif') right c
  }
  </style>
  <script>
  $(function() {
    function log( message ) {
      $( "<div>" ).text( message ).prependTo( "#log" );
      $( "#log" ).scrollTop( 0 );
    }

    $( "#birds" ).autocomplete({
      source: "search.php",
      minLength: 2,
      select: function( event, ui ) {
        log( ui.item ?
          "Selected: " + ui.item.value + " aka " + ui.item.id :
          "Nothing selected, input was " + this.value );
      }
    });
  });
  </script>
</head>
<body>

<div class="ui-widget">
  <label for="birds">鸟：</label>
  <input id="birds">
</div>

<div class="ui-widget" style="margin-top:2em; font-family:Arial">
  结果：
  <div id="log" style="height: 200px; width: 300px; overflow: auto;
</div>

</body>
</html>
```

远程缓存

当您在文本域中键入字符时，Autocomplete 部件给出建议结果。在本实例中，当您在文本域中至少键入两个字符时，将显示相关鸟的名称。

为了提高性能，这里添加了一些本地缓存，其他与远程数据源实例相似。在这里，缓存只保存了一个查询，并可以扩展到缓存多个值，每个条目一个值。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 自动完成（Autocomplete） - 远程缓存</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  .ui-autocomplete-loading {
    background: white url('images/ui-anim_basic_16x16.gif') right c
  }
  </style>
  <script>
  $(function() {
    var cache = {};
    $( "#birds" ).autocomplete({
      minLength: 2,
      source: function( request, response ) {
        var term = request.term;
        if ( term in cache ) {
          response( cache[ term ] );
          return;
        }

        $.getJSON( "search.php", request, function( data, status, x
          cache[ term ] = data;
          response( data );
        });
      }
    });
  });
  </script>
</head>
<body>

<div class="ui-widget">
  <label for="birds">鸟：</label>
  <input id="birds">
</div>

</body>
</html>
```

可滚动的结果

当显示一个长列表的选项时，您可以简单地为 autocomplete 菜单设置 max-height 来防止菜单显示太长。尝试键入 "a" 或 "s" 来获得一个可滚动的长列表的结果。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 自动完成（Autocomplete） - 可滚动的结果</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  .ui-autocomplete {
    max-height: 100px;
    overflow-y: auto;
    /* 防止水平滚动条 */
    overflow-x: hidden;
  }
  /* IE 6 不支持 max-height
   * 我们使用 height 代替，但是这会强制菜单总是显示为那个高度
   */
  * html .ui-autocomplete {
    height: 100px;
  }
  </style>
  <script>
  $(function() {
    var availableTags = [
      "ActionScript",
      "AppleScript",
      "Asp",
      "BASIC",
      "C",
      "C++",
      "Clojure",
      "COBOL",
      "ColdFusion",
      "Erlang",
      "Fortran",
      "Groovy",
      "Haskell",
      "Java",
      "JavaScript",
      "Lisp",
      "Perl",
      "PHP",
      "Python",
      "Ruby",
```

```
      "Scala",
      "Scheme"
    ];
    $( "#tags" ).autocomplete({
      source: availableTags
    });
  });
  </script>
</head>
<body>

<div class="ui-widget">
  <label for="tags">标签：</label>
  <input id="tags">
</div>

</body>
</html>
```

XML 数据

本实例演示如何获取一些 XML 数据，并使用 jQuery 的方法解析它，然后把它提供给 autocomplete 作为数据源。

本实例也可作为解析远程 XML 数据源的参考 - 解析在每次 source 回调请求时发生。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 自动完成（Autocomplete） - XML 数据</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  .ui-autocomplete-loading { background: white url('images/ui-anim_
  </style>
  <script>
  $(function() {
    function log( message ) {
      $( "<div/>" ).text( message ).prependTo( "#log" );
      $( "#log" ).attr( "scrollTop", 0 );
    }

    $.ajax({
      url: "london.xml",
      dataType: "xml",
```

```
        success: function( xmlResponse ) {
          var data = $( "geoname", xmlResponse ).map(function() {
            return {
              value: $( "name", this ).text() + ", " +
                ( $.trim( $( "countryName", this ).text() ) || "(unkn
              id: $( "geonameId", this ).text()
            };
          }).get();
          $( "#birds" ).autocomplete({
            source: data,
            minLength: 0,
            select: function( event, ui ) {
              log( ui.item ?
                "Selected: " + ui.item.value + ", geonameId: " + ui.i
                "Nothing selected, input was " + this.value );
            }
          });
        }
      });
    });
    </script>
  </head>
  <body>

  <div class="ui-widget">
    <label for="birds">London 匹配：</label>
    <input id="birds">
  </div>

  <div class="ui-widget" style="margin-top:2em; font-family:Arial">
    结果：
    <div id="log" style="height: 200px; width: 300px; overflow: auto;
  </div>

  </body>
  </html>
```

jQuery UI 实例 - 按钮（Button）

用带有适当的悬停（hover）和激活（active）的样式的可主题化按钮来加强标准表单元素（比如按钮、输入框、锚）的功能。

如需了解更多有关 button 部件的细节，请查看 API 文档 按钮部件（Button Widget）。

默认功能

可用于按钮的标记实例：一个 button 元素，一个类型为 submit 的 input 元素和一个锚。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 按钮（Button） - 默认功能</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "input[type=submit], a, button" )
      .button()
      .click(function( event ) {
        event.preventDefault();
      });
  });
  </script>
</head>
<body>

<button>一个 button 元素</button>

<input type="submit" value="一个提交按钮">

<a href="#">一个锚</a>

</body>
</html>
```

复选框

通过 button 部件把复选框显示为切换按钮样式。与复选框相关的 label 元素作为按钮文本。

本实例通过在一个公共的容器上调用 `.buttonset()` ，演示了三个显示为按钮样式的复选框。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 按钮（Button） - 复选框</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#check" ).button();
    $( "#format" ).buttonset();
  });
  </script>
  <style>
  #format { margin-top: 2em; }
  </style>
</head>
<body>

<input type="checkbox" id="check"><label for="check">切换</label>

<div id="format">
  <input type="checkbox" id="check1"><label for="check1">B</label>
  <input type="checkbox" id="check2"><label for="check2">I</label>
  <input type="checkbox" id="check3"><label for="check3">U</label>
</div>

</body>
</html>
```

图标

一些带有文本和图标的各种组合的按钮

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 按钮（Button） - 图标</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "button:first" ).button({
      icons: {
        primary: "ui-icon-locked"
      },
      text: false
    }).next().button({
      icons: {
        primary: "ui-icon-locked"
      }
    }).next().button({
      icons: {
        primary: "ui-icon-gear",
        secondary: "ui-icon-triangle-1-s"
      }
    }).next().button({
      icons: {
        primary: "ui-icon-gear",
        secondary: "ui-icon-triangle-1-s"
      },
      text: false
    });
  });
  </script>
</head>
<body>

<button>只带有图标的按钮</button>
<button>图标在左侧的按钮</button>
<button>带有两个图标的按钮</button>
<button>带有两个图标且不带文本的按钮</button>

</body>
</html>
```

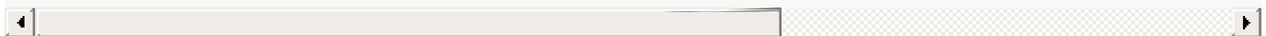
单选

三个单选按钮转变为一套按钮。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 按钮（Button） - 单选</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#radio" ).buttonset();
  });
  </script>
</head>
<body>

<form>
  <div id="radio">
    <input type="radio" id="radio1" name="radio"><label for="radio1
    <input type="radio" id="radio2" name="radio" checked="checked">
    <input type="radio" id="radio3" name="radio"><label for="radio3
  </div>
</form>

</body>
</html>
```



分割按钮

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 按钮（Button） - 分割按钮</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
    .ui-menu { position: absolute; width: 100px; }
  </style>
  <script>
  $(function() {
    $( "#rerun" )
      .button()
      .click(function() {
        alert( "Running the last action" );
```

```
      })
      .next()
        .button({
          text: false,
          icons: {
            primary: "ui-icon-triangle-1-s"
          }
        })
        .click(function() {
          var menu = $( this ).parent().next().show().position({
            my: "left top",
            at: "left bottom",
            of: this
          });
          $( document ).one( "click", function() {
            menu.hide();
          });
          return false;
        })
        .parent()
          .buttonset()
          .next()
            .hide()
            .menu();
  });
  </script>
</head>
<body>

<div>
  <div>
    <button id="rerun">运行最后的动作</button>
    <button id="select">选择一个动作</button>
  </div>
  <ul>
    <li><a href="#">打开...</a></li>
    <li><a href="#">保存</a></li>
    <li><a href="#">删除</a></li>
  </ul>
</div>

</body>
</html>
```

工具栏

一个多媒体播放器的工具栏。请看基础的标记：一些 button 元素，Shuffle 按钮是一个类型为 checkbox 的 input，Repeat 选项是三个类型为 radio 的 input。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 按钮（Button） - 工具栏</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #toolbar {
    padding: 4px;
    display: inline-block;
  }
  /* support: IE7 */
  *+html #toolbar {
    display: inline;
  }
  </style>
  <script>
  $(function() {
    $( "#beginning" ).button({
      text: false,
      icons: {
        primary: "ui-icon-seek-start"
      }
    });
    $( "#rewind" ).button({
      text: false,
      icons: {
        primary: "ui-icon-seek-prev"
      }
    });
    $( "#play" ).button({
      text: false,
      icons: {
        primary: "ui-icon-play"
      }
    })
    .click(function() {
      var options;
      if ( $( this ).text() === "play" ) {
        options = {
          label: "pause",
          icons: {
            primary: "ui-icon-pause"
          }
        };
      } else {
        options = {
          label: "play",
          icons: {
```

```
              primary: "ui-icon-play"
            }
          };
        }
        $( this ).button( "option", options );
      });
      $( "#stop" ).button({
        text: false,
        icons: {
          primary: "ui-icon-stop"
        }
      })
      .click(function() {
        $( "#play" ).button( "option", {
          label: "play",
          icons: {
            primary: "ui-icon-play"
          }
        });
      });
      $( "#forward" ).button({
        text: false,
        icons: {
          primary: "ui-icon-seek-next"
        }
      });
      $( "#end" ).button({
        text: false,
        icons: {
          primary: "ui-icon-seek-end"
        }
      });
      $( "#shuffle" ).button();
      $( "#repeat" ).buttonset();
    });
    </script>
  </head>
  <body>

  <div id="toolbar" class="ui-widget-header ui-corner-all">
    <button id="beginning">开头</button>
    <button id="rewind">快退</button>
    <button id="play">播放</button>
    <button id="stop">停止</button>
    <button id="forward">快进</button>
    <button id="end">结尾</button>

    <input type="checkbox" id="shuffle"><label for="shuffle">Shuffle<

    <span id="repeat">
      <input type="radio" id="repeat0" name="repeat" checked="checkec
      <input type="radio" id="repeat1" name="repeat"><label for="repe
      <input type="radio" id="repeatall" name="repeat"><label for="re
```

```
  </span>
</div>

</body>
</html>
```

'Attr.nodeValue' is deprecated. Please use 'value' instead. adsbygoogle.js:32 href http://www.w3cschool.cc/jqueryui/example-datepicker.html example-datepicker.html:438 tmp_url http://www.w3cschool.cc/jqueryui/example-datepicker.html example-datepicker.html:450 example-button.html jQuery UI 实例 – 按钮（Button） example-datepicker.html:495 example-dialog.html jQuery UI 实例 – 对话框（Dialog） example-datepicker.html:496

jQuery UI 实例 - 日期选择器（Datepicker）

从弹出框或内联日历选择一个日期。

如需了解更多有关 datepicker 部件的细节，请查看 API 文档 日期选择器部件（Datepicker Widget）。

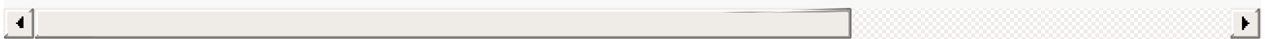
默认功能

日期选择器（Datepicker）绑定到一个标准的表单 input 字段上。把焦点移到 input 上（点击或者使用 tab 键），在一个小的覆盖层上打开一个交互日历。选择一个日期，点击页面上的任意地方（输入框即失去焦点），或者点击 Esc 键来关闭。如果选择了一个日期，则反馈显示为 input 的值。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 日期选择器（Datepicker） - 默认功能</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#datepicker" ).datepicker();
  });
  </script>
</head>
<body>

<p>日期：<input type="text" id="datepicker"></p>

</body>
</html>
```



动画

当打开或关闭 datepicker 时使用不同的动画。从下拉框中选择一个动画，然后在输入框中点击来查看它的效果。您可以使用三个标准动画中任意一个，或者使用 UI 特效中的任意一个。

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>jQuery UI 日期选择器（Datepicker） - 动画</title>
    <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes
    <script src="//code.jquery.com/jquery-1.9.1.js"></script>
    <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script
    <link rel="stylesheet" href="http://jqueryui.com/resources/demo
    <script>
    $(function() {
        $( "#datepicker" ).datepicker();
        $( "#anim" ).change(function() {
            $( "#datepicker" ).datepicker( "option", "showAnim", $(
        });
    });
    </script>
</head>
<body>

<p>日期：<input type="text" id="datepicker" size="30"></p>

<p>动画：<br>
    <select id="anim">
        <option value="show">Show (默认)</option>
        <option value="slideDown">滑下</option>
        <option value="fadeIn">淡入</option>
        <option value="blind">Blind (UI 百叶窗特效)</option>
        <option value="bounce">Bounce (UI 反弹特效)</option>
        <option value="clip">Clip (UI 剪辑特效)</option>
        <option value="drop">Drop (UI 降落特效)</option>
        <option value="fold">Fold (UI 折叠特效)</option>
        <option value="slide">Slide (UI 滑动特效)</option>
        <option value="">无</option>
    </select>
</p>

</body>
</html>
```

其他月份的日期

datepicker 可以显示其他月份的日期，这些日期也可以设置成可选择的。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 日期选择器（Datepicker） - 其他月份的日期</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#datepicker" ).datepicker({
      showOtherMonths: true,
      selectOtherMonths: true
    });
  });
  </script>
</head>
<body>

<p>日期：<input type="text" id="datepicker"></p>

</body>
</html>
```

显示按钮栏

通过布尔值的 `showButtonPanel` 选项为选择当天日期显示一个"Today"按钮，为关闭日历显示一个"Done"按钮。默认情况下，当按钮栏显示时会启用每个按钮，但是按钮可通过其他的选项进行关闭。按钮文本可自定义。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 日期选择器（Datepicker） - 显示按钮栏</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#datepicker" ).datepicker({
      showButtonPanel: true
    });
  });
  </script>
</head>
<body>

<p>日期：<input type="text" id="datepicker"></p>

</body>
</html>
```

内联显示

datepicker 是嵌套在页面中显示，而不是显示在一个覆盖层中。只需要简单地在 div 上调用 .datepicker() 即可，而不是在 input 上调用。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 日期选择器（Datepicker） - 内联显示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#datepicker" ).datepicker();
  });
  </script>
</head>
<body>

日期：<div id="datepicker"></div>

</body>
</html>
```

显示月份 & 年份菜单

显示月份和年份的下拉框，而不是显示静态的月份/年份标题，这样便于在大范围的时间跨度上导航。添加布尔值 `changeMonth` 和 `changeYear` 选项即可。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 日期选择器（Datepicker） - 显示月份 &amp; 年份菜单</t
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#datepicker" ).datepicker({
      changeMonth: true,
      changeYear: true
    });
  });
  </script>
</head>
<body>

<p>日期：<input type="text" id="datepicker"></p>

</body>
</html>
```

显示多个月份

设置 `numberOfMonths` 选项为一个整数 2，或者大于 2 的整数，来在一个 datepicker 中显示多个月份。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 日期选择器（Datepicker） - 显示多个月份</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#datepicker" ).datepicker({
      numberOfMonths: 3,
      showButtonPanel: true
    });
  });
  </script>
</head>
<body>

<p>日期：<input type="text" id="datepicker"></p>

</body>
</html>
```

格式化日期

以各种方式显示日期反馈。从下拉框中选择一种日期格式，然后在输入框中点击并选择一个日期，查看所选格式的日期显示。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 日期选择器（Datepicker） - 格式化日期</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#datepicker" ).datepicker();
    $( "#format" ).change(function() {
      $( "#datepicker" ).datepicker( "option", "dateFormat", $( thi
    });
  });
  </script>
</head>
<body>

<p>日期：<input type="text" id="datepicker" size="30"></p>

<p>格式选项：<br>
  <select id="format">
    <option value="mm/dd/yy">Default - mm/dd/yy</option>
    <option value="yy-mm-dd">ISO 8601 - yy-mm-dd</option>
    <option value="d M, y">Short - d M, y</option>
    <option value="d MM, y">Medium - d MM, y</option>
    <option value="DD, d MM, yy">Full - DD, d MM, yy</option>
    <option value="&apos;day&apos; d &apos;of&apos; MM &apos;in the
  </select>
</p>

</body>
</html>
```



图标触发器

点击输入框旁边的图标来显示 datepicker。设置 datepicker 在获得焦点时打开（默认行为），或者在点击图标时打开，或者在获得焦点/点击图标时打开。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 日期选择器（Datepicker） - 图标触发器</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#datepicker" ).datepicker({
      showOn: "button",
      buttonImage: "images/calendar.gif",
      buttonImageOnly: true
    });
  });
  </script>
</head>
<body>

<p>日期：<input type="text" id="datepicker"></p>

</body>
</html>
```

本地化日历

本地化 datepicker 日历语言和格式（默认为 English / Western 格式）。datepicker 包含对从右到左读取的语言的内建支持，比如 Arabic 和 Hebrew。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 日期选择器（Datepicker） - 本地化日历</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <script src="http://jqueryui.com/resources/demos/datepicker/jquer
  <script src="http://jqueryui.com/resources/demos/datepicker/jquer
  <script src="http://jqueryui.com/resources/demos/datepicker/jquer
  <script src="http://jqueryui.com/resources/demos/datepicker/jquer
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#datepicker" ).datepicker( $.datepicker.regional[ "fr" ] );
    $( "#locale" ).change(function() {
      $( "#datepicker" ).datepicker( "option",
        $.datepicker.regional[ $( this ).val() ] );
    });
  });
  </script>
</head>
<body>

<p>日期：<input type="text" id="datepicker"> 
  <select id="locale">
    <option value="ar">Arabic (&#8235;(&#1575;&#1604;&#1593;&#1585;
    <option value="zh-TW">Chinese Traditional (&#32321;&#39636;&#20
    <option value="">English</option>
    <option value="fr" selected="selected">French (Fran&ccedil;ais)
    <option value="he">Hebrew (&#8235;(&#1506;&#1489;&#1512;&#1497;
  </select></p>

</body>
</html>
```

填充另一个输入框

使用 `altField` 和 `altFormat` 选项，无论何时选择日期，会在另一个输入框中填充带有一定格式的日期。这个功能通过对电脑友好性的日期进一步加工后，向用户呈现一个用户友好性的日期。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 日期选择器（Datepicker） - 填充另一个输入框</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#datepicker" ).datepicker({
      altField: "#alternate",
      altFormat: "DD, d MM, yy"
    });
  });
  </script>
</head>
<body>

<p>日期：<input type="text" id="datepicker"> <input type="text"

</body>
</html>
```

限制日期范围

通过 `minDate` 和 `maxDate` 选项限制可选择的日期范围。设置起止日期为实际的日期（new Date(2009, 1 - 1, 26)），或者为与今天的一个数值偏移（-20），或者为一个周期和单位的字符串（'+1M +10D'）。如果设置为字符串，使用 'D' 表示天，使用 'W' 表示周，使用 'M' 表示月，使用 'Y' 表示年。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 日期选择器（Datepicker） - 限制日期范围</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#datepicker" ).datepicker({ minDate: -20, maxDate: "+1M +10
  });
  </script>
</head>
<body>

<p>日期：<input type="text" id="datepicker"></p>

</body>
</html>
```

选择一个日期范围

选择要搜索的日期范围。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 日期选择器（Datepicker） - 选择一个日期范围</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#from" ).datepicker({
      defaultDate: "+1w",
      changeMonth: true,
      numberOfMonths: 3,
      onClose: function( selectedDate ) {
        $( "#to" ).datepicker( "option", "minDate", selectedDate );
      }
    });
    $( "#to" ).datepicker({
      defaultDate: "+1w",
      changeMonth: true,
      numberOfMonths: 3,
      onClose: function( selectedDate ) {
        $( "#from" ).datepicker( "option", "maxDate", selectedDate
      }
    });
  });
  </script>
</head>
<body>

<label for="from">从</label>
<input type="text" id="from" name="from">
<label for="to">到</label>
<input type="text" id="to" name="to">

</body>
</html>
```

显示一年中的第几周

datepicker 可以显示一年中的第几周。默认的计算是按照 ISO 8601 定义：每周从星期一开始，每年的第一周包含该年的第一个星期四。这就意味着一年中的一些天可能是属于另一年中的周。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 日期选择器（Datepicker） - 显示一年中的第几周</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#datepicker" ).datepicker({
      showWeek: true,
      firstDay: 1
    });
  });
  </script>
</head>
<body>

<p>日期：<input type="text" id="datepicker"></p>

</body>
</html>
```

jQuery UI 实例 - 对话框（Dialog）

在一个交互覆盖层中打开内容。

如需了解更多有关 dialog 部件的细节，请查看 API 文档 对话框部件（Dialog Widget）。

默认功能

基本的对话框窗口是一个定位于视区中的覆盖层，同时通过一个 iframe 与页面内容分隔开（就像 select 元素）。它由一个标题栏和一个内容区域组成，且可以移动，调整尺寸，默认可通过 'x' 图标关闭。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 对话框（Dialog） - 默认功能</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#dialog" ).dialog();
  });
  </script>
</head>
<body>

<div id="dialog" title="基本的对话框">
  <p>这是一个默认的对话框，用于显示信息。对话框窗口可以移动，调整尺寸，默认可通
</div>

</body>
</html>
```

动画

可以通过为 show/hide 属性指定一个特效来动画显示对话框。您必须为想使用的特效引用独立的特效文件。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 对话框（Dialog） - 动画</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#dialog" ).dialog({
      autoOpen: false,
      show: {
        effect: "blind",
        duration: 1000
      },
      hide: {
        effect: "explode",
        duration: 1000
      }
    });

    $( "#opener" ).click(function() {
      $( "#dialog" ).dialog( "open" );
    });
  });
  </script>
</head>
<body>

<div id="dialog" title="Basic dialog">
  <p>这是一个动画显示的对话框，用于显示信息。对话框窗口可以移动，调整尺寸，默认
</div>

<button id="opener">打开对话框</button>

</body>
</html>
```

基本的模态

模态对话框防止用户与对话框以外的页面其他部分进行交互，直到对话框关闭为止。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 对话框（Dialog） - 基本的模态</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#dialog-modal" ).dialog({
      height: 140,
      modal: true
    });
  });
  </script>
</head>
<body>

<div id="dialog-modal" title="基本的模态对话框">
  <p>添加模态覆盖屏幕，让对话框看起来更突出，因为它让页面上其他内容变暗。</p>
</div>

<p>Sed vel diam id libero <a href="http://www.w3cschool.cc">rutrum

</body>
</html>
```

模态确认

确认一个动作可能是破坏性的也可能是有意义的。设置 `modal` 选项为 true，并通过 `buttons` 选项来指定主要的和次要的用户动作。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 对话框（Dialog） - 模态确认</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#dialog-confirm" ).dialog({
      resizable: false,
      height:140,
      modal: true,
      buttons: {
        "Delete all items": function() {
          $( this ).dialog( "close" );
        },
        Cancel: function() {
          $( this ).dialog( "close" );
        }
      }
    });
  });
  </script>
</head>
<body>

<div id="dialog-confirm" title="清空回收站吗？">
  <p><span class="ui-icon ui-icon-alert" style="float:left; margin
</div>

<p>Sed vel diam id libero <a href="http://www.w3cschool.cc">rutrum

</body>
</html>
```

模态表单

使用模态对话框来请求用户在一个多步骤过程中输入数据。在内容区域嵌入 form 标记，设置 `modal` 选项为 true，并通过 `buttons` 选项来指定主要的和次要的用户动作。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
```

```
  <title>jQuery UI 对话框（Dialog） - 模态表单</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
    body { font-size: 62.5%; }
    label, input { display:block; }
    input.text { margin-bottom:12px; width:95%; padding: .4em; }
    fieldset { padding:0; border:0; margin-top:25px; }
    h1 { font-size: 1.2em; margin: .6em 0; }
    div#users-contain { width: 350px; margin: 20px 0; }
    div#users-contain table { margin: 1em 0; border-collapse: colla
    div#users-contain table td, div#users-contain table th { border
    .ui-dialog .ui-state-error { padding: .3em; }
    .validateTips { border: 1px solid transparent; padding: 0.3em;
  </style>
  <script>
  $(function() {
    var name = $( "#name" ),
      email = $( "#email" ),
      password = $( "#password" ),
      allFields = $( [] ).add( name ).add( email ).add( password ),
      tips = $( ".validateTips" );

    function updateTips( t ) {
      tips
        .text( t )
        .addClass( "ui-state-highlight" );
      setTimeout(function() {
        tips.removeClass( "ui-state-highlight", 1500 );
      }, 500 );
    }

    function checkLength( o, n, min, max ) {
      if ( o.val().length > max || o.val().length < min ) {
        o.addClass( "ui-state-error" );
        updateTips( "" + n + " 的长度必须在 " +
          min + " 和 " + max + " 之间。" );
        return false;
      } else {
        return true;
      }
    }

    function checkRegexp( o, regexp, n ) {
      if ( !( regexp.test( o.val() ) ) ) {
        o.addClass( "ui-state-error" );
        updateTips( n );
        return false;
      } else {
        return true;
      }
```

```
    }

    $( "#dialog-form" ).dialog({
      autoOpen: false,
      height: 300,
      width: 350,
      modal: true,
      buttons: {
        "创建一个帐户": function() {
          var bValid = true;
          allFields.removeClass( "ui-state-error" );

          bValid = bValid && checkLength( name, "username", 3, 16 )
          bValid = bValid && checkLength( email, "email", 6, 80 );
          bValid = bValid && checkLength( password, "password", 5,

          bValid = bValid && checkRegexp( name, /^[a-z]([0-9a-z_])-
          // From jquery.validate.js (by joern), contributed by Sco
          bValid = bValid && checkRegexp( email, /^((([a-z]|\d|[!#`
          bValid = bValid && checkRegexp( password, /^([0-9a-zA-Z])

          if ( bValid ) {
            $( "#users tbody" ).append( "<tr>" +
              "<td>" + name.val() + "</td>" +
              "<td>" + email.val() + "</td>" +
              "<td>" + password.val() + "</td>" +
            "</tr>" );
            $( this ).dialog( "close" );
          }
        },
        Cancel: function() {
          $( this ).dialog( "close" );
        }
      },
      close: function() {
        allFields.val( "" ).removeClass( "ui-state-error" );
      }
    });

    $( "#create-user" )
      .button()
      .click(function() {
        $( "#dialog-form" ).dialog( "open" );
      });
  });
  </script>
</head>
<body>

<div id="dialog-form" title="创建新用户">
  <p class="validateTips">所有的表单字段都是必填的。</p>

  <form>
```

```
    <fieldset>
      <label for="name">名字</label>
      <input type="text" name="name" id="name" class="text ui-widget-
      <label for="email">邮箱</label>
      <input type="text" name="email" id="email" value="" class="text
      <label for="password">密码</label>
      <input type="password" name="password" id="password" value="" c
    </fieldset>
    </form>
  </div>

  <div id="users-contain" class="ui-widget">
    <h1>已有的用户：</h1>
    <table id="users" class="ui-widget ui-widget-content">
      <thead>
        <tr class="ui-widget-header ">
          <th>名字</th>
          <th>邮箱</th>
          <th>密码</th>
        </tr>
      </thead>
      <tbody>
        <tr>
          <td>John Doe</td>
          <td>john.doe@example.com</td>
          <td>johndoe1</td>
        </tr>
      </tbody>
    </table>
  </div>
  <button id="create-user">创建新用户</button>

  </body>
  </html>
```

模态消息

使用模态对话框来在下一步动作执行之前确认信息和动作。设置 `modal` 选项为 true，并通过 `buttons` 选项来指定主要的动作（Ok）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 对话框（Dialog） - 模态消息</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#dialog-message" ).dialog({
      modal: true,
      buttons: {
        Ok: function() {
          $( this ).dialog( "close" );
        }
      }
    });
  });
  </script>
</head>
<body>

<div id="dialog-message" title="下载完成">
  <p>
    <span class="ui-icon ui-icon-circle-check" style="float:left; m
    您的文件已经成功下载到文件夹中。
  </p>
  <p>
    当前使用存储空间的 <b>36%</b>。
  </p>
</div>

<p>Sed vel diam id libero <a href="http://www.w3cschool.cc">rutrum

</body>
</html>
```

jQuery UI 实例 - 菜单（Menu）

带有鼠标和键盘交互的用于导航的可主题化菜单。

如需了解更多有关 menu 部件的细节，请查看 API 文档 菜单部件（Menu Widget）。

默认功能

一个带有默认配置、禁用条目和嵌套菜单的菜单。由一个列表转化成的，添加了主题，并支持鼠标和键盘交互。尝试使用光标键导航菜单。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 菜单（Menu） - 默认功能</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#menu" ).menu();
  });
  </script>
  <style>
  .ui-menu { width: 150px; }
  </style>
</head>
<body>

<ul id="menu">
  <li class="ui-state-disabled"><a href="#">Aberdeen</a></li>
  <li><a href="#">Ada</a></li>
  <li><a href="#">Adamsville</a></li>
  <li><a href="#">Addyston</a></li>
  <li>
    <a href="#">Delphi</a>
    <ul>
      <li class="ui-state-disabled"><a href="#">Ada</a></li>
      <li><a href="#">Saarland</a></li>
      <li><a href="#">Salzburg</a></li>
    </ul>
  </li>
  <li><a href="#">Saarland</a></li>
  <li>
    <a href="#">Salzburg</a>
```

```
    <ul>
      <li>
        <a href="#">Delphi</a>
        <ul>
          <li><a href="#">Ada</a></li>
          <li><a href="#">Saarland</a></li>
          <li><a href="#">Salzburg</a></li>
        </ul>
      </li>
      <li>
        <a href="#">Delphi</a>
        <ul>
          <li><a href="#">Ada</a></li>
          <li><a href="#">Saarland</a></li>
          <li><a href="#">Salzburg</a></li>
        </ul>
      </li>
      <li><a href="#">Perch</a></li>
    </ul>
  </li>
  <li class="ui-state-disabled"><a href="#">Amesville</a></li>
</ul>

</body>
</html>
```

图标

一个带有默认配置的菜单，显示如何使用带有图标的菜单。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 菜单（Menu） - 图标</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#menu" ).menu();
  });
  </script>
  <style>
  .ui-menu { width: 150px; }
  </style>
</head>
<body>

<ul id="menu">
  <li><a href="#"><span class="ui-icon ui-icon-disk"></span>保存</a
  <li><a href="#"><span class="ui-icon ui-icon-zoomin"></span>放大<
  <li><a href="#"><span class="ui-icon ui-icon-zoomout"></span>缩小
  <li class="ui-state-disabled"><a href="#"><span class="ui-icon ui
  <li>
    <a href="#">播放</a>
    <ul>
      <li><a href="#"><span class="ui-icon ui-icon-seek-start"></sp
      <li><a href="#"><span class="ui-icon ui-icon-stop"></span>停止
      <li><a href="#"><span class="ui-icon ui-icon-play"></span>播放
      <li><a href="#"><span class="ui-icon ui-icon-seek-end"></span
    </ul>
  </li>
</ul>

</body>
</html>
```

jQuery UI 实例 - 进度条（Progressbar）

显示一个确定的或不确定的进程状态。

如需了解更多有关 progressbar 部件的细节，请查看 API 文档 进度条部件（Progressbar Widget）。

默认功能

默认的确定的进度条。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 进度条（Progressbar） - 默认功能</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#progressbar" ).progressbar({
      value: 37
    });
  });
  </script>
</head>
<body>

<div id="progressbar"></div>

</body>
</html>
```

自定义标签

自定义更新的标签。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 进度条（Progressbar） - 自定义标签</title>
```

```
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  .ui-progressbar {
    position: relative;
  }
  .progress-label {
    position: absolute;
    left: 50%;
    top: 4px;
    font-weight: bold;
    text-shadow: 1px 1px 0 #fff;
  }
  </style>
  <script>
  $(function() {
    var progressbar = $( "#progressbar" ),
      progressLabel = $( ".progress-label" );

    progressbar.progressbar({
      value: false,
      change: function() {
        progressLabel.text( progressbar.progressbar( "value" ) + "%
      },
      complete: function() {
        progressLabel.text( "完成！" );
      }
    });

    function progress() {
      var val = progressbar.progressbar( "value" ) || 0;

      progressbar.progressbar( "value", val + 1 );

      if ( val < 99 ) {
        setTimeout( progress, 100 );
      }
    }

    setTimeout( progress, 3000 );
  });
  </script>
</head>
<body>

<div id="progressbar"><div class="progress-label">加载...</div></di

</body>
</html>
```

不确定的值

不确定的进度条，并可在确定和不确定的样式之间切换。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 进度条（Progressbar） - 不确定的值</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#progressbar" ).progressbar({
      value: false
    });
    $( "button" ).on( "click", function( event ) {
      var target = $( event.target ),
        progressbar = $( "#progressbar" ),
        progressbarValue = progressbar.find( ".ui-progressbar-value

      if ( target.is( "#numButton" ) ) {
        progressbar.progressbar( "option", {
          value: Math.floor( Math.random() * 100 )
        });
      } else if ( target.is( "#colorButton" ) ) {
        progressbarValue.css({
          "background": '#' + Math.floor( Math.random() * 16777215
        });
      } else if ( target.is( "#falseButton" ) ) {
        progressbar.progressbar( "option", "value", false );
      }
    });
  });
  </script>
  <style>
  #progressbar .ui-progressbar-value {
    background-color: #ccc;
  }
  </style>
</head>
<body>

<div id="progressbar"></div>
<button id="numButton">随机值 - 确定</button>
<button id="falseButton">不确定</button>
<button id="colorButton">随机颜色</button>

</body>
</html>
```

jQuery UI 实例 - 滑块（Slider）

拖动手柄来选择一个数值。

如需了解更多有关 slider 部件的细节，请查看 API 文档 滑块部件（Slider Widget）。

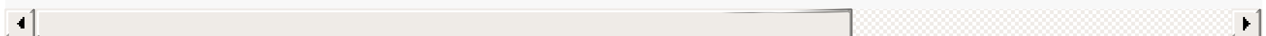
默认功能

基本的滑块是水平的，有一个单一的手柄，可以用鼠标或箭头键进行移动。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 滑块（Slider） - 默认功能</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#slider" ).slider();
  });
  </script>
</head>
<body>

<div id="slider"></div>

</body>
</html>
```



颜色选择器

组合了三个滑块，来创建一个简单的 RGB 颜色选择器。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 滑块（Slider） - 颜色选择器</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
```

```
    <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
    <style>
    #red, #green, #blue {
      float: left;
      clear: left;
      width: 300px;
      margin: 15px;
    }
    #swatch {
      width: 120px;
      height: 100px;
      margin-top: 18px;
      margin-left: 350px;
      background-image: none;
    }
    #red .ui-slider-range { background: #ef2929; }
    #red .ui-slider-handle { border-color: #ef2929; }
    #green .ui-slider-range { background: #8ae234; }
    #green .ui-slider-handle { border-color: #8ae234; }
    #blue .ui-slider-range { background: #729fcf; }
    #blue .ui-slider-handle { border-color: #729fcf; }
    </style>
    <script>
    function hexFromRGB(r, g, b) {
      var hex = [
        r.toString( 16 ),
        g.toString( 16 ),
        b.toString( 16 )
      ];
      $.each( hex, function( nr, val ) {
        if ( val.length === 1 ) {
          hex[ nr ] = "0" + val;
        }
      });
      return hex.join( "" ).toUpperCase();
    }
    function refreshSwatch() {
      var red = $( "#red" ).slider( "value" ),
        green = $( "#green" ).slider( "value" ),
        blue = $( "#blue" ).slider( "value" ),
        hex = hexFromRGB( red, green, blue );
      $( "#swatch" ).css( "background-color", "#" + hex );
    }
    $(function() {
      $( "#red, #green, #blue" ).slider({
        orientation: "horizontal",
        range: "min",
        max: 255,
        value: 127,
        slide: refreshSwatch,
        change: refreshSwatch
      });
```

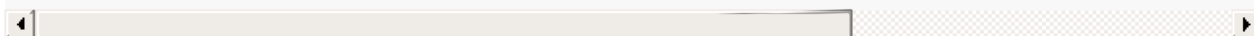
```
    $( "#red" ).slider( "value", 255 );
    $( "#green" ).slider( "value", 140 );
    $( "#blue" ).slider( "value", 60 );
  });
  </script>
</head>
<body class="ui-widget-content" style="border:0;">

<p class="ui-state-default ui-corner-all ui-helper-clearfix" style=
  <span class="ui-icon ui-icon-pencil" style="float:left; margin:-2
  简单的颜色选择器
</p>

<div id="red"></div>
<div id="green"></div>
<div id="blue"></div>

<div id="swatch" class="ui-widget-content ui-corner-all"></div>

</body>
</html>
```



多个滑块

组合水平的和垂直的滑块，每个都带有各自的选项，来创建一个音乐播放器的 UI。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 滑块（Slider） - 多个滑块</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #eq span {
    height:120px; float:left; margin:15px
  }
  </style>
  <script>
  $(function() {
    // 设置主音量
    $( "#master" ).slider({
      value: 60,
      orientation: "horizontal",
      range: "min",
      animate: true
    });
```

```
    // 设置图形均衡器
    $( "#eq > span" ).each(function() {
      // 从标记读取初始值并删除
      var value = parseInt( $( this ).text(), 10 );
      $( this ).empty().slider({
        value: value,
        range: "min",
        animate: true,
        orientation: "vertical"
      });
    });
  });
  </script>
</head>
<body>

<p class="ui-state-default ui-corner-all ui-helper-clearfix" style=
  <span class="ui-icon ui-icon-volume-on" style="float:left; margin
  主音量
</p>

<div id="master" style="width:260px; margin:15px;"></div>

<p class="ui-state-default ui-corner-all" style="padding:4px;margin
  <span class="ui-icon ui-icon-signal" style="float:left; margin:-2
  图形均衡器
</p>

<div id="eq">
  <span>88</span>
  <span>77</span>
  <span>55</span>
  <span>33</span>
  <span>40</span>
  <span>45</span>
  <span>70</span>
</div>

</body>
</html>
```

范围滑块

设置 `range` 选项为 true，来获取带有两个拖拽手柄的值的范围。手柄之间的控件用不同的背景颜色填充来表示该区间的值是可选择的。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 滑块（Slider） - 范围滑块</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#slider-range" ).slider({
      range: true,
      min: 0,
      max: 500,
      values: [ 75, 300 ],
      slide: function( event, ui ) {
        $( "#amount" ).val( "$" + ui.values[ 0 ] + " - $" + ui.valu
      }
    });
    $( "#amount" ).val( "$" + $( "#slider-range" ).slider( "values"
      " - $" + $( "#slider-range" ).slider( "values", 1 ) );
  });
  </script>
</head>
<body>

<p>
  <label for="amount">价格范围：</label>
  <input type="text" id="amount" style="border:0; color:#f6931f; fo
</p>

<div id="slider-range"></div>

</body>
</html>
```

带有固定最大值的范围

固定范围滑块的最大值，用户只能选择最小值。设置 `range` 选项为 "max"。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 滑块（Slider） - 带有固定最大值的范围</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#slider-range-max" ).slider({
      range: "max",
      min: 1,
      max: 10,
      value: 2,
      slide: function( event, ui ) {
        $( "#amount" ).val( ui.value );
      }
    });
    $( "#amount" ).val( $( "#slider-range-max" ).slider( "value" )
  });
  </script>
</head>
<body>

<p>
  <label for="amount">最小的房间数量：</label>
  <input type="text" id="amount" style="border:0; color:#f6931f; fo
</p>
<div id="slider-range-max"></div>

</body>
</html>
```

带有固定最小值的范围

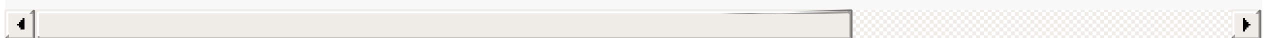
固定范围滑块的最小值，用户只能选择最大值。设置 `range` 选项为 "min"。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 滑块（Slider） - 带有固定最小值的范围</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#slider-range-min" ).slider({
      range: "min",
      value: 37,
      min: 1,
      max: 700,
      slide: function( event, ui ) {
        $( "#amount" ).val( "$" + ui.value );
      }
    });
    $( "#amount" ).val( "$" + $( "#slider-range-min" ).slider( "val
  });
  </script>
</head>
<body>

<p>
  <label for="amount">最大价格：</label>
  <input type="text" id="amount" style="border:0; color:#f6931f; fo
</p>

<div id="slider-range-min"></div>

</body>
</html>
```



绑定到 select 的滑块

如何绑定一个滑块到一个已有的 select 元素。选择保持可见以便显示变化。当选择改变时，同时更新滑块。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 滑块（Slider） - 绑定到 select 的滑块</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    var select = $( "#minbeds" );
    var slider = $( "<div id='slider'></div>" ).insertAfter( select
      min: 1,
      max: 6,
      range: "min",
      value: select[ 0 ].selectedIndex + 1,
      slide: function( event, ui ) {
        select[ 0 ].selectedIndex = ui.value - 1;
      }
    });
    $( "#minbeds" ).change(function() {
      slider.slider( "value", this.selectedIndex + 1 );
    });
  });
  </script>
</head>
<body>

<form id="reservation">
  <label for="minbeds">最小的床位数</label>
  <select name="minbeds" id="minbeds">
    <option>1</option>
    <option>2</option>
    <option>3</option>
    <option>4</option>
    <option>5</option>
    <option>6</option>
  </select>
</form>

</body>
</html>
```

滑块滚动条

使用滑块来操作页面上内容的定位。本实例中，它是一个能获取值的滚动条。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 滑块（Slider） - 滑块滚动条</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  .scroll-pane { overflow: auto; width: 99%; float:left; }
  .scroll-content { width: 2440px; float: left; }
  .scroll-content-item { width: 100px; height: 100px; float: left;
  .scroll-bar-wrap { clear: left; padding: 0 4px 0 2px; margin: 0 -
  .scroll-bar-wrap .ui-slider { background: none; border:0; height
  .scroll-bar-wrap .ui-handle-helper-parent { position: relative; w
  .scroll-bar-wrap .ui-slider-handle { top:.2em; height: 1.5em; }
  .scroll-bar-wrap .ui-slider-handle .ui-icon { margin: -8px auto 0
  </style>
  <script>
  $(function() {
    //滚动面板部分
    var scrollPane = $( ".scroll-pane" ),
      scrollContent = $( ".scroll-content" );

    //创建滑块
    var scrollbar = $( ".scroll-bar" ).slider({
      slide: function( event, ui ) {
        if ( scrollContent.width() > scrollPane.width() ) {
          scrollContent.css( "margin-left", Math.round(
            ui.value / 100 * ( scrollPane.width() - scrollContent.w
          ) + "px" );
        } else {
          scrollContent.css( "margin-left", 0 );
        }
      }
    });

    //追加要处理的图标
    var handleHelper = scrollbar.find( ".ui-slider-handle" )
    .mousedown(function() {
      scrollbar.width( handleHelper.width() );
    })
    .mouseup(function() {
      scrollbar.width( "100%" );
    })
    .append( "<span class='ui-icon ui-icon-grip-dotted-vertical'></
    .wrap( "<div class='ui-handle-helper-parent'></div>" ).parent()

    //由于滑块手柄滚动，改变要隐藏的溢出部分
    scrollPane.css( "overflow", "hidden" );
```

```
    //根据要滚动距离按比例定义滚动条和手柄的尺寸
    function sizeScrollbar() {
      var remainder = scrollContent.width() - scrollPane.width();
      var proportion = remainder / scrollContent.width();
      var handleSize = scrollPane.width() - ( proportion * scrollPa
      scrollbar.find( ".ui-slider-handle" ).css({
        width: handleSize,
        "margin-left": -handleSize / 2
      });
      handleHelper.width( "" ).width( scrollbar.width() - handleSiz
    }

    //基于滚动内容位置，重置滑块的值
    function resetValue() {
      var remainder = scrollPane.width() - scrollContent.width();
      var leftVal = scrollContent.css( "margin-left" ) === "auto" ?
        parseInt( scrollContent.css( "margin-left" ) );
      var percentage = Math.round( leftVal / remainder * 100 );
      scrollbar.slider( "value", percentage );
    }

    //如果滑块是 100%，且窗口变大，则显示内容
    function reflowContent() {
        var showing = scrollContent.width() + parseInt( scrollConte
        var gap = scrollPane.width() - showing;
        if ( gap > 0 ) {
          scrollContent.css( "margin-left", parseInt( scrollContent
        }
    }

    //当缩放窗口时改变手柄的位置
    $( window ).resize(function() {
      resetValue();
      sizeScrollbar();
      reflowContent();
    });
    //初始化滚动条大小
    setTimeout( sizeScrollbar, 10 );//safari 超时
  });
  </script>
</head>
<body>

<div class="scroll-pane ui-widget ui-widget-header ui-corner-all">
  <div class="scroll-content">
    <div class="scroll-content-item ui-widget-header">1</div>
    <div class="scroll-content-item ui-widget-header">2</div>
    <div class="scroll-content-item ui-widget-header">3</div>
    <div class="scroll-content-item ui-widget-header">4</div>
    <div class="scroll-content-item ui-widget-header">5</div>
    <div class="scroll-content-item ui-widget-header">6</div>
    <div class="scroll-content-item ui-widget-header">7</div>
    <div class="scroll-content-item ui-widget-header">8</div>
```

```
    <div class="scroll-content-item ui-widget-header">9</div>
    <div class="scroll-content-item ui-widget-header">10</div>
    <div class="scroll-content-item ui-widget-header">11</div>
    <div class="scroll-content-item ui-widget-header">12</div>
    <div class="scroll-content-item ui-widget-header">13</div>
    <div class="scroll-content-item ui-widget-header">14</div>
    <div class="scroll-content-item ui-widget-header">15</div>
    <div class="scroll-content-item ui-widget-header">16</div>
    <div class="scroll-content-item ui-widget-header">17</div>
    <div class="scroll-content-item ui-widget-header">18</div>
    <div class="scroll-content-item ui-widget-header">19</div>
    <div class="scroll-content-item ui-widget-header">20</div>
  </div>
  <div class="scroll-bar-wrap ui-widget-content ui-corner-bottom">
    <div class="scroll-bar"></div>
  </div>
</div>

</body>
</html>
```

对齐增量

通过把 `step` 选项设置为一个整数来设置滑块值的增量，通常是滑块最大值的除数。默认增量是 1。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 滑块（Slider） - 对齐增量</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#slider" ).slider({
      value:100,
      min: 0,
      max: 500,
      step: 50,
      slide: function( event, ui ) {
        $( "#amount" ).val( "$" + ui.value );
      }
    });
    $( "#amount" ).val( "$" + $( "#slider" ).slider( "value" ) );
  });
  </script>
</head>
<body>

<p>
  <label for="amount">捐款金额（$50 增量）：</label>
  <input type="text" id="amount" style="border:0; color:#f6931f; fo
</p>

<div id="slider"></div>

</body>
</html>
```

垂直的范围滑块

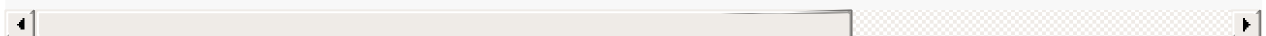
改变范围滑块的方向为垂直的。通过 `.height()` 分配一个高度值，或通过 CSS 设置高度，同时设置 `orientation` 选项为 "vertical"。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 滑块（Slider） - 垂直的范围滑块</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#slider-range" ).slider({
      orientation: "vertical",
      range: true,
      values: [ 17, 67 ],
      slide: function( event, ui ) {
        $( "#amount" ).val( "$" + ui.values[ 0 ] + " - $" + ui.valu
      }
    });
    $( "#amount" ).val( "$" + $( "#slider-range" ).slider( "values"
      " - $" + $( "#slider-range" ).slider( "values", 1 ) );
  });
  </script>
</head>
<body>

<p>
  <label for="amount">销售目标（百万）：</label>
  <input type="text" id="amount" style="border:0; color:#f6931f; fo
</p>

<div id="slider-range" style="height:250px;"></div>

</body>
</html>
```



垂直的滑块

改变滑块的方向为垂直的。通过 `.height()` 分配一个高度值，或通过 CSS 设置高度，同时设置 `orientation` 选项为 "vertical"。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 滑块（Slider） - 垂直的滑块</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#slider-vertical" ).slider({
      orientation: "vertical",
      range: "min",
      min: 0,
      max: 100,
      value: 60,
      slide: function( event, ui ) {
        $( "#amount" ).val( ui.value );
      }
    });
    $( "#amount" ).val( $( "#slider-vertical" ).slider( "value" ) )
  });
  </script>
</head>
<body>

<p>
  <label for="amount">体积：</label>
  <input type="text" id="amount" style="border:0; color:#f6931f; fo
</p>

<div id="slider-vertical" style="height:200px;"></div>

</body>
</html>
```

jQuery UI 实例 - 旋转器（Spinner）

通过向上/向下按钮和箭头键处理，为输入数值增强文本输入功能。

如需了解更多有关 spinner 部件的细节，请查看 API 文档 旋转器部件（Spinner Widget）。

默认功能

默认的旋转器。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 旋转器（Spinner） - 默认功能</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="http://jqueryui.com/resources/demos/external/jquery.
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    var spinner = $( "#spinner" ).spinner();

    $( "#disable" ).click(function() {
      if ( spinner.spinner( "option", "disabled" ) ) {
        spinner.spinner( "enable" );
      } else {
        spinner.spinner( "disable" );
      }
    });
    $( "#destroy" ).click(function() {
      if ( spinner.data( "ui-spinner" ) ) {
        spinner.spinner( "destroy" );
      } else {
        spinner.spinner();
      }
    });
    $( "#getvalue" ).click(function() {
      alert( spinner.spinner( "value" ) );
    });
    $( "#setvalue" ).click(function() {
      spinner.spinner( "value", 5 );
    });

    $( "button" ).button();
  });
```

```
  </script>
</head>
<body>

<p>
  <label for="spinner">选择一个值：</label>
  <input id="spinner" name="value">
</p>

<p>
  <button id="disable">切换禁用/启用</button>
  <button id="destroy">切换部件</button>
</p>

<p>
  <button id="getvalue">获取值</button>
  <button id="setvalue">设置值为 5</button>
</p>

</body>
</html>
```

货币

本实例是一个捐款表格，带有货币选择和数量旋转器。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 旋转器（Spinner） - 货币</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="http://jqueryui.com/resources/demos/external/jquery.
  <script src="http://jqueryui.com/resources/demos/external/globali
  <script src="http://jqueryui.com/resources/demos/external/globali
  <script src="http://jqueryui.com/resources/demos/external/globali
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#currency" ).change(function() {
      $( "#spinner" ).spinner( "option", "culture", $( this ).val()
    });

    $( "#spinner" ).spinner({
      min: 5,
      max: 2500,
      step: 25,
      start: 1000,
      numberFormat: "C"
    });
  });
  </script>
</head>
<body>

<p>
  <label for="currency">要捐款的货币</label>
  <select id="currency" name="currency">
    <option value="en-US">US $</option>
    <option value="de-DE">EUR €</option>
    <option value="ja-JP">YEN ¥</option>
  </select>
</p>
<p>
  <label for="spinner">要捐款的数量：</label>
  <input id="spinner" name="spinner" value="5">
</p>

</body>
</html>
```

小数

本实例是一个小数旋转器。增量设置为 0.01。处理文化变化的代码会读取当前的选择器的值，当改变文化时，会基于新的文化重新设置值的样式。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 旋转器（Spinner） - 小数</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="http://jqueryui.com/resources/demos/external/jquery.
  <script src="http://jqueryui.com/resources/demos/external/globali
  <script src="http://jqueryui.com/resources/demos/external/globali
  <script src="http://jqueryui.com/resources/demos/external/globali
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#spinner" ).spinner({
      step: 0.01,
      numberFormat: "n"
    });

    $( "#culture" ).change(function() {
      var current = $( "#spinner" ).spinner( "value" );
      Globalize.culture( $(this).val() );
      $( "#spinner" ).spinner( "value", current );
    });
  });
  </script>
</head>
<body>

<p>
  <label for="spinner">小数旋转器：</label>
  <input id="spinner" name="spinner" value="5.06">
</p>
<p>
  <label for="culture">选择一种用于格式化的文化：</label>
  <select id="culture">
    <option value="en-EN" selected="selected">English</option>
    <option value="de-DE">German</option>
    <option value="ja-JP">Japanese</option>
  </select>
</p>

</body>
</html>
```

地图

谷歌地图集成，使用旋转器来改变纬度和经度。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 旋转器（Spinner） - 地图</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="http://maps.google.com/maps/api/js?sensor=false"></s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="http://jqueryui.com/resources/demos/external/jquery.
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    function latlong() {
      return new google.maps.LatLng( $("#lat").val(), $("#lng").val
    }
    function position() {
      map.setCenter( latlong() );
    }
    $( "#lat, #lng" ).spinner({
      step: .001,
      change: position,
      stop: position
    });

    var map = new google.maps.Map( $("#map")[0], {
      zoom: 8,
      center: latlong(),
      mapTypeId: google.maps.MapTypeId.ROADMAP
    });
  });
  </script>
  <style>
  #map {
    width:500px;
    height:500px;
  }
  </style>
</head>
<body>

<label for="lat">纬度</label>
<input id="lat" name="lat" value="44.797">
<br>
<label for="lng">经度</label>
<input id="lng" name="lng" value="-93.278">
```

```
<div id="map"></div>

</body>
</html>
```

溢出

溢出旋转器限制范围从 -10 到 10。对于 10 以上的值，会溢出到 -10，反之亦然。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 旋转器（Spinner） - 溢出</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="http://jqueryui.com/resources/demos/external/jquery.
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#spinner" ).spinner({
      spin: function( event, ui ) {
        if ( ui.value > 10 ) {
          $( this ).spinner( "value", -10 );
          return false;
        } else if ( ui.value < -10 ) {
          $( this ).spinner( "value", 10 );
          return false;
        }
      }
    });
  });
  </script>
</head>
<body>

<p>
  <label for="spinner">选择一个值：</label>
  <input id="spinner" name="value">
</p>

</body>
</html>
```

时间

一个扩展自旋转器的自定义部件。使用 全球化（Globalization）插件来解析和输出时间戳，带有自定义的 step 和 page 选项。向上/向下光标用于分钟的递增/递减，向上/向下翻页用于小时的递增/递减。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 旋转器（Spinner） - 时间</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="http://jqueryui.com/resources/demos/external/jquery.
  <script src="http://jqueryui.com/resources/demos/external/globali
  <script src="http://jqueryui.com/resources/demos/external/globali
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $.widget( "ui.timespinner", $.ui.spinner, {
    options: {
      // 秒
      step: 60 * 1000,
      // 小时
      page: 60
    },

    _parse: function( value ) {
      if ( typeof value === "string" ) {
        // 已经是一个时间戳
        if ( Number( value ) == value ) {
          return Number( value );
        }
        return +Globalize.parseDate( value );
      }
      return value;
    },

    _format: function( value ) {
      return Globalize.format( new Date(value), "t" );
    }
  });

  $(function() {
    $( "#spinner" ).timespinner();

    $( "#culture" ).change(function() {
      var current = $( "#spinner" ).timespinner( "value" );
      Globalize.culture( $(this).val() );
      $( "#spinner" ).timespinner( "value", current );
    });
  });
  </script>
</head>
```

```
<body>

<p>
  <label for="spinner">时间旋转器：</label>
  <input id="spinner" name="spinner" value="08:30 PM">
</p>
<p>
  <label for="culture">选择一种用于格式化的文化：</label>
  <select id="culture">
    <option value="en-EN" selected="selected">English</option>
    <option value="de-DE">German</option>
  </select>
</p>

</body>
</html>
```

jQuery UI 实例 - 标签页（Tabs）

一种多面板的单内容区，每个面板与列表中的标题相关。

如需了解更多有关 tabs 部件的细节，请查看 API 文档 标签页部件（Tabs Widget）。

默认功能

点击标签页，切换被划分为不同逻辑部分的内容。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 标签页（Tabs） - 默认功能</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#tabs" ).tabs();
  });
  </script>
</head>
<body>

<div id="tabs">
  <ul>
    <li><a href="#tabs-1">Nunc tincidunt</a></li>
    <li><a href="#tabs-2">Proin dolor</a></li>
    <li><a href="#tabs-3">Aenean lacinia</a></li>
  </ul>
  <div id="tabs-1">
    <p>Proin elit arcu, rutrum commodo, vehicula tempus, commodo a,
  </div>
  <div id="tabs-2">
    <p>Morbi tincidunt, dui sit amet facilisis feugiat, odio metus
  </div>
  <div id="tabs-3">
    <p>Mauris eleifend est et turpis. Duis id erat. Suspendisse pot
    <p>Duis cursus. Maecenas ligula eros, blandit nec, pharetra at,
  </div>
</div>

</body>
</html>
```

折叠内容

点击选中的标签页来切换内容的关闭/打开状态。为了启用这个功能，需要设置 `collapsible` 选项为 true。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 标签页（Tabs） - 折叠内容</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#tabs" ).tabs({
      collapsible: true
    });
  });
  </script>
</head>
<body>

<div id="tabs">
  <ul>
    <li><a href="#tabs-1">Nunc tincidunt</a></li>
    <li><a href="#tabs-2">Proin dolor</a></li>
    <li><a href="#tabs-3">Aenean lacinia</a></li>
  </ul>
  <div id="tabs-1">
    <p><strong>再次点击标签页来关闭内容面板。</strong></p>
    <p>Proin elit arcu, rutrum commodo, vehicula tempus, commodo a,
  </div>
  <div id="tabs-2">
    <p><strong>再次点击标签页来关闭内容面板。</strong></p>
    <p>Morbi tincidunt, dui sit amet facilisis feugiat, odio metus
  </div>
  <div id="tabs-3">
    <p><strong>再次点击标签页来关闭内容面板。</strong></p>
    <p>Duis cursus. Maecenas ligula eros, blandit nec, pharetra at,
  </div>
</div>

</body>
</html>
```

通过 Ajax 获取内容

在标签页链接中设置 href 的值来为标签页通过 Ajax 获取外部的内容。当 Ajax 请求在等待响应时，标签页的标签变为 "Loading..."，当加载完成后返回常规的标签。

标签 3 和 4 演示了慢加载和损坏的 AJAX 标签，以及如何处理这些情况下的服务器端的错误。请注意，这两个都要求 web 服务器能解释 PHP。它们在文件系统以外无法工作。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 标签页（Tabs） - 通过 Ajax 获取内容</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#tabs" ).tabs({
      beforeLoad: function( event, ui ) {
        ui.jqXHR.error(function() {
          ui.panel.html(
            "不能加载该标签页。如果这不是一个演示。" +
            "我们会尽快修复这个问题。" );
        });
      }
    });
  });
  </script>
</head>
<body>

<div id="tabs">
  <ul>
    <li><a href="#tabs-1">预加载</a></li>
    <li><a href="ajax/content1.html">标签 1</a></li>
    <li><a href="ajax/content2.html">标签 2</a></li>
    <li><a href="ajax/content3-slow.php">标签 3 （慢的）</a></li>
    <li><a href="ajax/content4-broken.php">标签 4 （损坏的）</a></li>
  </ul>
  <div id="tabs-1">
    <p>Proin elit arcu, rutrum commodo, vehicula tempus, commodo a,
  </div>
</div>

</body>
</html>
```



当鼠标悬停时打开

通过 `event` 选项设置当鼠标悬停时切换各部分的打开/关闭状态。event 的默认值是 "click"。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 标签页（Tabs） - 当鼠标悬停时打开</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#tabs" ).tabs({
      event: "mouseover"
    });
  });
  </script>
</head>
<body>

<div id="tabs">
  <ul>
    <li><a href="#tabs-1">Nunc tincidunt</a></li>
    <li><a href="#tabs-2">Proin dolor</a></li>
    <li><a href="#tabs-3">Aenean lacinia</a></li>
  </ul>
  <div id="tabs-1">
    <p>Proin elit arcu, rutrum commodo, vehicula tempus, commodo a,
  </div>
  <div id="tabs-2">
    <p>Morbi tincidunt, dui sit amet facilisis feugiat, odio metus
  </div>
  <div id="tabs-3">
    <p>Mauris eleifend est et turpis. Duis id erat. Suspendisse pot
    <p>Duis cursus. Maecenas ligula eros, blandit nec, pharetra at,
  </div>
</div>

</body>
</html>
```



简单的操作

简单的标签页添加和移除。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
```

```
<title>jQuery UI 标签页（Tabs） - 简单的操作</title>
<link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
<script src="//code.jquery.com/jquery-1.9.1.js"></script>
<script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
<link rel="stylesheet" href="http://jqueryui.com/resources/demos/
<style>
#dialog label, #dialog input { display:block; }
#dialog label { margin-top: 0.5em; }
#dialog input, #dialog textarea { width: 95%; }
#tabs { margin-top: 1em; }
#tabs li .ui-icon-close { float: left; margin: 0.4em 0.2em 0 0; c
#add_tab { cursor: pointer; }
</style>
<script>
$(function() {
  var tabTitle = $( "#tab_title" ),
    tabContent = $( "#tab_content" ),
    tabTemplate = "<li><a href='#{href}'>#{label}</a> <span class
    tabCounter = 2;

  var tabs = $( "#tabs" ).tabs();

  // 模态对话框的初始化：自定义按钮和一个重置内部表单的 "close" 回调
  var dialog = $( "#dialog" ).dialog({
    autoOpen: false,
    modal: true,
    buttons: {
      Add: function() {
        addTab();
        $( this ).dialog( "close" );
      },
      Cancel: function() {
        $( this ).dialog( "close" );
      }
    },
    close: function() {
      form[ 0 ].reset();
    }
  });

  // addTab 表单：当提交时调用 addTab 函数，并关闭对话框
  var form = dialog.find( "form" ).submit(function( event ) {
    addTab();
    dialog.dialog( "close" );
    event.preventDefault();
  });

  // 实际的 addTab 函数：使用上面表单的输入添加新的标签页
  function addTab() {
    var label = tabTitle.val() || "Tab " + tabCounter,
      id = "tabs-" + tabCounter,
      li = $( tabTemplate.replace( /#\{href\}/g, "#" + id ).repla
      tabContentHtml = tabContent.val() || "Tab " + tabCounter +
```

```
      tabs.find( ".ui-tabs-nav" ).append( li );
      tabs.append( "<div id='" + id + "'><p>" + tabContentHtml + "<
      tabs.tabs( "refresh" );
      tabCounter++;
    }

    // addTab 按钮：值打开对话框
    $( "#add_tab" )
      .button()
      .click(function() {
        dialog.dialog( "open" );
      });

    // 关闭图标：当点击时移除标签页
    tabs.delegate( "span.ui-icon-close", "click", function() {
      var panelId = $( this ).closest( "li" ).remove().attr( "aria-
      $( "#" + panelId ).remove();
      tabs.tabs( "refresh" );
    });

    tabs.bind( "keyup", function( event ) {
      if ( event.altKey && event.keyCode === $.ui.keyCode.BACKSPACE
        var panelId = tabs.find( ".ui-tabs-active" ).remove().attr(
        $( "#" + panelId ).remove();
        tabs.tabs( "refresh" );
      }
    });
  });
  </script>
</head>
<body>

<div id="dialog" title="Tab data">
  <form>
    <fieldset class="ui-helper-reset">
      <label for="tab_title">标题</label>
      <input type="text" name="tab_title" id="tab_title" value="" c
      <label for="tab_content">内容</label>
      <textarea name="tab_content" id="tab_content" class="ui-widge
    </fieldset>
  </form>
</div>

<button id="add_tab">添加标签页</button>

<div id="tabs">
  <ul>
    <li><a href="#tabs-1">Nunc tincidunt</a> <span class="ui-icon u
  </ul>
  <div id="tabs-1">
    <p>Proin elit arcu, rutrum commodo, vehicula tempus, commodo a,
  </div>
```

```
</div>

</body>
</html>
```

排序（Sortable）

拖拽上面的标签页来对它们进行重新排序。

只需要简单地调用 `.ui-tabs-nav` 元素上的 `.sortable()`，即可让标签页可排序。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 标签页（Tabs） - 排序（Sortable）</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    var tabs = $( "#tabs" ).tabs();
    tabs.find( ".ui-tabs-nav" ).sortable({
      axis: "x",
      stop: function() {
        tabs.tabs( "refresh" );
      }
    });
  });
  </script>
</head>
<body>

<div id="tabs">
  <ul>
    <li><a href="#tabs-1">Nunc tincidunt</a></li>
    <li><a href="#tabs-2">Proin dolor</a></li>
    <li><a href="#tabs-3">Aenean lacinia</a></li>
  </ul>
  <div id="tabs-1">
    <p>Proin elit arcu, rutrum commodo, vehicula tempus, commodo a,
  </div>
  <div id="tabs-2">
    <p>Morbi tincidunt, dui sit amet facilisis feugiat, odio metus
  </div>
  <div id="tabs-3">
    <p>Mauris eleifend est et turpis. Duis id erat. Suspendisse pot
    <p>Duis cursus. Maecenas ligula eros, blandit nec, pharetra at,
  </div>
</div>

</body>
</html>
```



底部标签页

通过一些额外的 CSS（用于定位的）和 JS（在元素上放置正确的 class），标签页皆可放置在内容的底部。

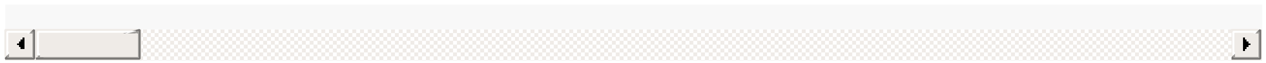
```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 标签页（Tabs） - 底部标签页</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#tabs" ).tabs();

    // 修改 class
    $( ".tabs-bottom .ui-tabs-nav, .tabs-bottom .ui-tabs-nav > *" )
      .removeClass( "ui-corner-all ui-corner-top" )
      .addClass( "ui-corner-bottom" );

    // 移动导航到底部
    $( ".tabs-bottom .ui-tabs-nav" ).appendTo( ".tabs-bottom" );
  });
  </script>
  <style>
  /* 强制一个高度，这样标签页就不会随着内容高度的改变而改变 */
  #tabs .tabs-spacer { float: left; height: 200px; }
  .tabs-bottom .ui-tabs-nav { clear: left; padding: 0 .2em .2em .2e
  .tabs-bottom .ui-tabs-nav li { top: auto; bottom: 0; margin: 0 .2
  .tabs-bottom .ui-tabs-nav li.ui-tabs-active { margin-top: -1px; p
  </style>
</head>
<body>

<div id="tabs" class="tabs-bottom">
  <ul>
    <li><a href="#tabs-1">Nunc tincidunt</a></li>
    <li><a href="#tabs-2">Proin dolor</a></li>
    <li><a href="#tabs-3">Aenean lacinia</a></li>
  </ul>
  <div class="tabs-spacer"></div>
  <div id="tabs-1">
    <p>Proin elit arcu, rutrum commodo, vehicula tempus, commodo a,
  </div>
  <div id="tabs-2">
    <p>Morbi tincidunt, dui sit amet facilisis feugiat, odio metus
  </div>
  <div id="tabs-3">
    <p>Mauris eleifend est et turpis. Duis id erat. Suspendisse pot
    <p>Duis cursus. Maecenas ligula eros, blandit nec, pharetra at,
  </div>
</div>

</body>
</html>
```



垂直的标签页

点击标签页，切换被划分为不同逻辑部分的内容。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 标签页（Tabs） - 垂直的标签页</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#tabs" ).tabs().addClass( "ui-tabs-vertical ui-helper-clear
    $( "#tabs li" ).removeClass( "ui-corner-top" ).addClass( "ui-co
  });
  </script>
  <style>
  .ui-tabs-vertical { width: 55em; }
  .ui-tabs-vertical .ui-tabs-nav { padding: .2em .1em .2em .2em; fl
  .ui-tabs-vertical .ui-tabs-nav li { clear: left; width: 100%; bor
  .ui-tabs-vertical .ui-tabs-nav li a { display:block; }
  .ui-tabs-vertical .ui-tabs-nav li.ui-tabs-active { padding-bottom
  .ui-tabs-vertical .ui-tabs-panel { padding: 1em; float: right; wi
  </style>
</head>
<body>

<div id="tabs">
  <ul>
    <li><a href="#tabs-1">Nunc tincidunt</a></li>
    <li><a href="#tabs-2">Proin dolor</a></li>
    <li><a href="#tabs-3">Aenean lacinia</a></li>
  </ul>
  <div id="tabs-1">
    <h2>内容标题 1</h2>
    <p>Proin elit arcu, rutrum commodo, vehicula tempus, commodo a,
  </div>
  <div id="tabs-2">
    <h2>内容标题 2</h2>
    <p>Morbi tincidunt, dui sit amet facilisis feugiat, odio metus
  </div>
  <div id="tabs-3">
    <h2>内容标题 3</h2>
    <p>Mauris eleifend est et turpis. Duis id erat. Suspendisse pot
    <p>Duis cursus. Maecenas ligula eros, blandit nec, pharetra at,
  </div>
</div>

</body>
</html>
```



jQuery UI 实例 - 工具提示框（Tooltip）

可自定义的、可主题化的工具提示框，替代原生的工具提示框。

如需了解更多有关 tooltip 部件的细节，请查看 API 文档 工具提示框部件（Tooltip Widget）。

默认功能

悬停在链接上，或者使用 tab 键循环切换聚焦在每个元素上。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 工具提示框（Tooltip） - 默认功能</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( document ).tooltip();
  });
  </script>
  <style>
  label {
    display: inline-block;
    width: 5em;
  }
  </style>
</head>
<body>

<p><a href="#" title="部件的名称">Tooltips</a> 可被绑定到任意的元素上。当
<p>但是由于它不是一个原生的工具提示框，所以它可以被定义样式。通过 <a href="ht
<p>工具提示框也可以用于表单元素，来显示每个区域中的一些额外的信息。</p>
<p><label for="age">您的年龄：</label><input id="age" title="年龄仅用于
<p>悬停在相应的区域上查看工具提示框。</p>

</body>
</html>
```



自定义样式

悬停在链接上，或者使用 tab 键循环切换聚焦在每个元素上。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 工具提示框（Tooltip） - 自定义样式</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( document ).tooltip({
      position: {
        my: "center bottom-20",
        at: "center top",
        using: function( position, feedback ) {
          $( this ).css( position );
          $( "<div>" )
            .addClass( "arrow" )
            .addClass( feedback.vertical )
            .addClass( feedback.horizontal )
            .appendTo( this );
        }
      }
    });
  });
  </script>
  <style>
  .ui-tooltip, .arrow:after {
    background: black;
    border: 2px solid white;
  }
  .ui-tooltip {
    padding: 10px 20px;
    color: white;
    border-radius: 20px;
    font: bold 14px "Helvetica Neue", Sans-Serif;
    text-transform: uppercase;
    box-shadow: 0 0 7px black;
  }
  .arrow {
    width: 70px;
    height: 16px;
    overflow: hidden;
    position: absolute;
    left: 50%;
    margin-left: -35px;
    bottom: -16px;
  }
  .arrow.top {
```

```
      top: -16px;
      bottom: auto;
    }
    .arrow.left {
      left: 20%;
    }
    .arrow:after {
      content: "";
      position: absolute;
      left: 20px;
      top: -20px;
      width: 25px;
      height: 25px;
      box-shadow: 6px 5px 9px -9px black;
      -webkit-transform: rotate(45deg);
      -moz-transform: rotate(45deg);
      -ms-transform: rotate(45deg);
      -o-transform: rotate(45deg);
      tranform: rotate(45deg);
    }
    .arrow.top:after {
      bottom: -20px;
      top: auto;
    }
    </style>
  </head>
  <body>

  <p><a href="#" title="部件的名称">Tooltips</a> 可被绑定到任意的元素上。当
  <p>但是由于它不是一个原生的工具提示框，所以它可以被定义样式。通过 <a href="ht
  <p>工具提示框也可以用于表单元素，来显示每个区域中的一些额外的信息。</p>
  <p><label for="age">您的年龄：</label><input id="age" title="年龄仅用于
  <p>悬停在相应的区域上查看工具提示框。</p>

  </body>
  </html>
```

自定义动画

本实例演示了如何使用 show 和 hide 选项来自定义动画，也可以使用 open 事件来自定义动画。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 工具提示框（Tooltip） - 自定义动画</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#show-option" ).tooltip({
      show: {
        effect: "slideDown",
        delay: 250
      }
    });
    $( "#hide-option" ).tooltip({
      hide: {
        effect: "explode",
        delay: 250
      }
    });
    $( "#open-event" ).tooltip({
      show: null,
      position: {
        my: "left top",
        at: "left bottom"
      },
      open: function( event, ui ) {
        ui.tooltip.animate({ top: ui.tooltip.position().top + 10 },
      }
    });
  });
  </script>
</head>
<body>

<p>这里有多种方式自定义工具提示框的动画。</p>
<p>您可以使用 <a id="show-option" href="http://jqueryui.com/demos/to
<a id="hide-option" href="http://jqueryui.com/demos/tooltip/#option
<p>您也可以使用 <a id="open-event" href="http://jqueryui.com/demos/to

</body>
</html>
```

自定义内容

演示如何通过自定义 items 和 content 选项，来组合不同的事件委托工具提示框到一个单一的实例中。

您可能需要与地图工具提示框进行交互，这是未来版本中的一个待实现的功能。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 工具提示框（Tooltip） - 自定义内容</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  .photo {
    width: 300px;
    text-align: center;
  }
  .photo .ui-widget-header {
    margin: 1em 0;
  }
  .map {
    width: 350px;
    height: 350px;
  }
  .ui-tooltip {
    max-width: 350px;
  }
  </style>
  <script>
  $(function() {
    $( document ).tooltip({
      items: "img, [data-geo], [title]",
      content: function() {
        var element = $( this );
        if ( element.is( "[data-geo]" ) ) {
          var text = element.text();
          return "<img class='map' alt='" + text +
            "' src='http://maps.google.com/maps/api/staticmap?" +
            "zoom=11&size=350x350&maptype=terrain&sensor=false&cent
            text + "'>";
        }
        if ( element.is( "[title]" ) ) {
          return element.attr( "title" );
        }
        if ( element.is( "img" ) ) {
          return element.attr( "alt" );
        }
      }
    });
  });
```

```
  </script>
</head>
<body>

<div class="ui-widget photo">
  <div class="ui-widget-header ui-corner-all">
    <h2>圣史蒂芬大教堂（St. Stephen's Cathedral）</h2>
    <h3><a href="http://maps.google.com/maps?q=vienna,+austria&amp;
  </div>
  <a href="http://en.wikipedia.org/wiki/File:Wien_Stefansdom_DSC026
    <img src="img/st-stephens.jpg" alt="圣史蒂芬大教堂（St. Stephen&a
  </a>
</div>

<div class="ui-widget photo">
  <div class="ui-widget-header ui-corner-all">
    <h2>塔桥（Tower Bridge）</h2>
    <h3><a href="http://maps.google.com/maps?q=london,+england&amp;
  </div>
  <a href="http://en.wikipedia.org/wiki/File:Tower_bridge_London_Tw
    <img src="img/tower-bridge.jpg" alt="塔桥（Tower Bridge）" class
  </a>
</div>

<p>所有的图片源自 <a href="http://commons.wikimedia.org/wiki/Main_Pag

</body>
</html>
```



表单

使用下面的按钮来显示帮助文本，或者只需要让鼠标悬停在输入框上或者让输入框获得焦点，即可显示相应输入框的帮助文本。

在 CSS 中定义一个固定的宽度，让同时显示所有的帮助文本时看起来更一致。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 工具提示框（Tooltip） - 表单</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  label {
    display: inline-block; width: 5em;
  }
```

```
  fieldset div {
    margin-bottom: 2em;
  }
  fieldset .help {
    display: inline-block;
  }
  .ui-tooltip {
    width: 210px;
  }
  </style>
  <script>
  $(function() {
    var tooltips = $( "[title]" ).tooltip();
    $( "<button>" )
      .text( "Show help" )
      .button()
      .click(function() {
        tooltips.tooltip( "open" );
      })
      .insertAfter( "form" );
  });
  </script>
</head>
<body>

<form>
  <fieldset>
    <div>
      <label for="firstname">名字</label>
      <input id="firstname" name="firstname" title="请提供您的名字。"
    </div>
    <div>
      <label for="lastname">姓氏</label>
      <input id="lastname" name="lastname" title="请提供您的姓氏。">
    </div>
    <div>
      <label for="address">地址</label>
      <input id="address" name="address" title="您的家庭或工作地址。">
    </div>
  </fieldset>
</form>

</body>
</html>
```

跟踪鼠标

本实例中的工具提示框是相对于鼠标进行定位的，当鼠标在元素上移动时，它会跟随鼠标移动。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 工具提示框（Tooltip） - 跟踪鼠标</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  label {
    display: inline-block;
    width: 5em;
  }
  </style>
  <script>
  $(function() {
    $( document ).tooltip({
      track: true
    });
  });
  </script>
</head>
<body>

<p><a href="#" title="部件的名称">Tooltips</a> 可被绑定到任意的元素上。当
<p>但是由于它不是一个原生的工具提示框，所以它可以被定义样式。通过 <a href="ht
<p>工具提示框也可以用于表单元素，来显示每个区域中的一些额外的信息。</p>
<p><label for="age">您的年龄：</label><input id="age" title="年龄仅用于
<p>悬停在相应的区域上查看工具提示框。</p>

</body>
</html>
```



视频播放器

一个虚拟的视频播放器，带有喜欢/分享/统计按钮，每个按钮都带有自定义样式的工具提示框。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 工具提示框（Tooltip） - 视频播放器</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
```

```
<style>
.player {
  width: 500px;
  height: 300px;
  border: 2px groove gray;
  background: rgb(200, 200, 200);
  text-align: center;
  line-height: 300px;
}
.ui-tooltip {
  border: 1px solid white;
  background: rgba(20, 20, 20, 1);
  color: white;
}
.set {
  display: inline-block;
}
.notification {
  position: absolute;
  display: inline-block;
  font-size: 2em;
  padding: .5em;
  box-shadow: 2px 2px 5px -2px rgba(0,0,0,0.5);
}
</style>
<script>
$(function() {
  function notify( input ) {
    var msg = "已选择 " + $.trim( input.data( "tooltip-title" ) |
    $( "<div>" )
      .appendTo( document.body )
      .text( msg )
      .addClass( "notification ui-state-default ui-corner-bottom'
      .position({
        my: "center top",
        at: "center top",
        of: window
      })
      .show({
        effect: "blind"
      })
      .delay( 1000 )
      .hide({
        effect: "blind",
        duration: "slow"
      }, function() {
        $( this ).remove();
      });
  }

  $( "button" ).each(function() {
    var button = $( this ).button({
      icons: {
```

```
          primary: $( this ).data( "icon" )
        },
        text: !!$( this ).attr( "title" )
      });
      button.click(function() {
        notify( button );
      });
    });
    $( ".set" ).buttonset({
      items: "button"
    });

    $( document ).tooltip({
      position: {
        my: "center top",
        at: "center bottom+5",
      },
      show: {
        duration: "fast"
      },
      hide: {
        effect: "hide"
      }
    });
  });
  </script>
</head>
<body>

<div class="player">这里是视频 (HTML5?)</div>
<div class="tools">
  <span class="set">
    <button data-icon="ui-icon-circle-arrow-n" title="我喜欢这个视频"
    <button data-icon="ui-icon-circle-arrow-s">我不喜欢这个视频</butt
  </span>
  <div class="set">
    <button data-icon="ui-icon-circle-plus" title="添加到播放列表">添
    <button class="menu" data-icon="ui-icon-triangle-1-s">添加到收藏
  </div>
  <button title="分享这个视频">分享</button>
  <button data-icon="ui-icon-alert">标记为不恰当</button>
</div>

</body>
</html>
```

jQuery UI 实例 - 特效（Effect）

对一个元素应用动画特效。

如需了解更多有关 `.effect()` 方法的细节，请查看 API 文档 .effect()。

.effect() 演示

点击按钮预览特效。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 特效 - .effect() 演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
    .toggler { width: 500px; height: 200px; position: relative; }
    #button { padding: .5em 1em; text-decoration: none; }
    #effect { width: 240px; height: 135px; padding: 0.4em; position
    #effect h3 { margin: 0; padding: 0.4em; text-align: center; }
    .ui-effects-transfer { border: 2px dotted gray; }
  </style>
  <script>
  $(function() {
    // 运行当前选中的特效
    function runEffect() {
      // 从中获取特效类型
      var selectedEffect = $( "#effectTypes" ).val();

      // 大多数的特效类型默认不需要传递选项
      var options = {};
      // 一些特效带有必需的参数
      if ( selectedEffect === "scale" ) {
        options = { percent: 0 };
      } else if ( selectedEffect === "transfer" ) {
        options = { to: "#button", className: "ui-effects-transfer"
      } else if ( selectedEffect === "size" ) {
        options = { to: { width: 200, height: 60 } };
      }

      // 运行特效
      $( "#effect" ).effect( selectedEffect, options, 500, callback
    };
```

```
    // 回调函数
    function callback() {
      setTimeout(function() {
        $( "#effect" ).removeAttr( "style" ).hide().fadeIn();
      }, 1000 );
    };

    // 根据选择菜单值设置特效
    $( "#button" ).click(function() {
      runEffect();
      return false;
    });
  });
  </script>
</head>
<body>

<div class="toggler">
  <div id="effect" class="ui-widget-content ui-corner-all">
    <h3 class="ui-widget-header ui-corner-all">特效（Effect）</h3>
    <p>
      Etiam libero neque, luctus a, eleifend nec, semper at, lorem.
    </p>
  </div>
</div>

<select name="effects" id="effectTypes">
  <option value="blind">百叶窗特效（Blind Effect）</option>
  <option value="bounce">反弹特效（Bounce Effect）</option>
  <option value="clip">剪辑特效（Clip Effect）</option>
  <option value="drop">降落特效（Drop Effect）</option>
  <option value="explode">爆炸特效（Explode Effect）</option>
  <option value="fade">淡入淡出特效（Fade Effect）</option>
  <option value="fold">折叠特效（Fold Effect）</option>
  <option value="highlight">突出特效（Highlight Effect）</option>
  <option value="puff">膨胀特效（Puff Effect）</option>
  <option value="pulsate">跳动特效（Pulsate Effect）</option>
  <option value="scale">缩放特效（Scale Effect）</option>
  <option value="shake">震动特效（Shake Effect）</option>
  <option value="size">尺寸特效（Size Effect）</option>
  <option value="slide">滑动特效（Slide Effect）</option>
  <option value="transfer">转移特效（Transfer Effect）</option>
</select>

<a href="#" id="button" class="ui-state-default ui-corner-all">运行

</body>
</html>
```

Easing 演示

本实例使用 HTML Canvas 元素，绘制了 jQuery UI 提供的所有 easings。 点击每个图可查看该 easing 的行为。。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 特效 - Easing 演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  .graph {
    float: left;
    margin-left: 10px;
  }
  </style>
  <script>
  $(function() {
    if ( !$( "<canvas>" )[0].getContext ) {
      $( "<div>" ).text(
        "您的浏览器不支持 canvas，本演示需要在支持 canvas 的浏览器下进行。"
      ).appendTo( "#graphs" );
      return;
    }

    var i = 0,
      width = 100,
      height = 100;

    $.each( $.easing, function( name, impl ) {
      var graph = $( "<div>" ).addClass( "graph" ).appendTo( "#grap
        text = $( "<div>" ).text( ++i + ". " + name ).appendTo( gra
        wrap = $( "<div>" ).appendTo( graph ).css( 'overflow', 'hic
        canvas = $( "<canvas>" ).appendTo( wrap )[ 0 ];

      canvas.width = width;
      canvas.height = height;
      var drawHeight = height * 0.8,
        cradius = 10;
        ctx = canvas.getContext( "2d" );
      ctx.fillStyle = "black";

      // 绘制背景
      ctx.beginPath();
      ctx.moveTo( cradius, 0 );
      ctx.quadraticCurveTo( 0, 0, 0, cradius );
      ctx.lineTo( 0, height - cradius );
      ctx.quadraticCurveTo( 0, height, cradius, height );
      ctx.lineTo( width - cradius, height );
      ctx.quadraticCurveTo( width, height, width, height - cradius
```

```
        ctx.lineTo( width, 0 );
        ctx.lineTo( cradius, 0 );
        ctx.fill();

        // 绘制底线
        ctx.strokeStyle = "#555";
        ctx.beginPath();
        ctx.moveTo( width * 0.1, drawHeight + .5 );
        ctx.lineTo( width * 0.9, drawHeight + .5 );
        ctx.stroke();

        // 绘制顶线
        ctx.strokeStyle = "#555";
        ctx.beginPath();
        ctx.moveTo( width * 0.1, drawHeight * .3 - .5 );
        ctx.lineTo( width * 0.9, drawHeight * .3 - .5 );
        ctx.stroke();

        // 绘制 easing
        ctx.strokeStyle = "white";
        ctx.beginPath();
        ctx.lineWidth = 2;
        ctx.moveTo( width * 0.1, drawHeight );
        $.each( new Array( width ), function( position ) {
          var state = position / width,
            val = impl( state, position, 0, 1, width );
          ctx.lineTo( position * 0.8 + width * 0.1,
            drawHeight - drawHeight * val * 0.7 );
        });
        ctx.stroke();

        // 点击时动态改变
        graph.click(function() {
          wrap
            .animate( { height: "hide" }, 2000, name )
            .delay( 800 )
            .animate( { height: "show" }, 2000, name );
        });

        graph.width( width ).height( height + text.height() + 10 );
      });
    });
    </script>
  </head>
  <body>

  <div id="graphs"></div>

  </body>
  </html>
```

jQuery UI 实例 - 显示（Show）

使用自定义效果来显示匹配的元素。

如需了解更多有关 `.show()` 方法的细节，请查看 API 文档 .show()。

.show() 演示

点击按钮预览特效。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 特效 - .show() 演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  .toggler { width: 500px; height: 200px; }
  #button { padding: .5em 1em; text-decoration: none; }
  #effect { width: 240px; height: 135px; padding: 0.4em; position:
  #effect h3 { margin: 0; padding: 0.4em; text-align: center; }
  </style>
  <script>
  $(function() {
    // 运行当前选中的特效
    function runEffect() {
      // 从中获取特效类型
      var selectedEffect = $( "#effectTypes" ).val();

      // 大多数的特效类型默认不需要传递选项
      var options = {};
      // 一些特效带有必需的参数
      if ( selectedEffect === "scale" ) {
        options = { percent: 100 };
      } else if ( selectedEffect === "size" ) {
        options = { to: { width: 280, height: 185 } };
      }

      // 运行特效
      $( "#effect" ).show( selectedEffect, options, 500, callback )
    };

    // 回调函数
    function callback() {
      setTimeout(function() {
```

```
        $( "#effect:visible" ).removeAttr( "style" ).fadeOut();
      }, 1000 );
    };

    // 根据选择菜单值设置特效
    $( "#button" ).click(function() {
      runEffect();
      return false;
    });

    $( "#effect" ).hide();
  });
  </script>
</head>
<body>

<div class="toggler">
  <div id="effect" class="ui-widget-content ui-corner-all">
    <h3 class="ui-widget-header ui-corner-all">显示（Show）</h3>
    <p>
      Etiam libero neque, luctus a, eleifend nec, semper at, lorem.
    </p>
  </div>
</div>

 <select name="effects" id="effectTypes">
  <option value="blind">百叶窗特效（Blind Effect）</option>
  <option value="bounce">反弹特效（Bounce Effect）</option>
  <option value="clip">剪辑特效（Clip Effect）</option>
  <option value="drop">降落特效（Drop Effect）</option>
  <option value="explode">爆炸特效（Explode Effect）</option>
  <option value="fold">折叠特效（Fold Effect）</option>
  <option value="highlight">突出特效（Highlight Effect）</option>
  <option value="puff">膨胀特效（Puff Effect）</option>
  <option value="pulsate">跳动特效（Pulsate Effect）</option>
  <option value="scale">缩放特效（Scale Effect）</option>
  <option value="shake">震动特效（Shake Effect）</option>
  <option value="size">尺寸特效（Size Effect）</option>
  <option value="slide">滑动特效（Slide Effect）</option>
</select>

<a href="#" id="button" class="ui-state-default ui-corner-all">运行

</body>
</html>
```

jQuery UI 实例 - 隐藏（Hide）

使用自定义效果来隐藏匹配的元素。

如需了解更多有关 `.hide()` 方法的细节，请查看 API 文档 .hide()。

.hide() 演示

点击按钮预览特效。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 特效 - .hide() 演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  .toggler { width: 500px; height: 200px; }
  #button { padding: .5em 1em; text-decoration: none; }
  #effect { width: 240px; height: 135px; padding: 0.4em; position:
  #effect h3 { margin: 0; padding: 0.4em; text-align: center; }
  </style>
  <script>
  $(function() {
    // 运行当前选中的特效
    function runEffect() {
      // 从中获取特效类型
      var selectedEffect = $( "#effectTypes" ).val();

      // 大多数的特效类型默认不需要传递选项
      var options = {};
      // 一些特效带有必需的参数
      if ( selectedEffect === "scale" ) {
        options = { percent: 0 };
      } else if ( selectedEffect === "size" ) {
        options = { to: { width: 200, height: 60 } };
      }

      // 运行特效
      $( "#effect" ).hide( selectedEffect, options, 1000, callback
    };

    // 回调函数
    function callback() {
      setTimeout(function() {
```

```
        $( "#effect" ).removeAttr( "style" ).hide().fadeIn();
      }, 1000 );
    };

    // 根据选择菜单值设置特效
    $( "#button" ).click(function() {
      runEffect();
      return false;
    });
  });
  </script>
</head>
<body>

<div class="toggler">
  <div id="effect" class="ui-widget-content ui-corner-all">
    <h3 class="ui-widget-header ui-corner-all">隐藏（Hide）</h3>
    <p>
      Etiam libero neque, luctus a, eleifend nec, semper at, lorem.
    </p>
  </div>
</div>

<select name="effects" id="effectTypes">
  <option value="blind">百叶窗特效（Blind Effect）</option>
  <option value="bounce">反弹特效（Bounce Effect）</option>
  <option value="clip">剪辑特效（Clip Effect）</option>
  <option value="drop">降落特效（Drop Effect）</option>
  <option value="explode">爆炸特效（Explode Effect）</option>
  <option value="fold">折叠特效（Fold Effect）</option>
  <option value="highlight">突出特效（Highlight Effect）</option>
  <option value="puff">膨胀特效（Puff Effect）</option>
  <option value="pulsate">跳动特效（Pulsate Effect）</option>
  <option value="scale">缩放特效（Scale Effect）</option>
  <option value="shake">震动特效（Shake Effect）</option>
  <option value="size">尺寸特效（Size Effect）</option>
  <option value="slide">滑动特效（Slide Effect）</option>
</select>

<a href="#" id="button" class="ui-state-default ui-corner-all">运行

</body>
</html>
```

jQuery UI 实例 - 切换（Toggle）

使用自定义效果来显示或隐藏匹配的元素。

如需了解更多有关 `.toggle()` 方法的细节，请查看 API 文档 .toggle()。

.toggle() 演示

点击按钮预览特效。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 特效 - .toggle() 演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  .toggler {
    width: 500px;
    height: 200px;
  }
  #button {
    padding: .5em 1em;
    text-decoration: none;
  }
  #effect {
    position: relative;
    width: 240px;
    height: 135px;
    padding: 0.4em;
  }
  #effect h3 {
    margin: 0;
    padding: 0.4em;
    text-align: center;
  }
  </style>
  <script>
  $(function() {
    // 运行当前选中的特效
    function runEffect() {
      // 从中获取特效类型
      var selectedEffect = $( "#effectTypes" ).val();

      // 大多数的特效类型默认不需要传递选项
```

```
      var options = {};
      // 一些特效带有必需的参数
      if ( selectedEffect === "scale" ) {
        options = { percent: 0 };
      } else if ( selectedEffect === "size" ) {
        options = { to: { width: 200, height: 60 } };
      }

      // 运行特效
      $( "#effect" ).toggle( selectedEffect, options, 500 );
    };

    // 根据选择菜单值设置特效
    $( "#button" ).click(function() {
      runEffect();
      return false;
    });
  });
  </script>
</head>
<body>

<div class="toggler">
  <div id="effect" class="ui-widget-content ui-corner-all">
    <h3 class="ui-widget-header ui-corner-all">切换（Toggle）</h3>
    <p>
      Etiam libero neque, luctus a, eleifend nec, semper at, lorem.
    </p>
  </div>
</div>

<select name="effects" id="effectTypes">
  <option value="blind">百叶窗特效（Blind Effect）</option>
  <option value="bounce">反弹特效（Bounce Effect）</option>
  <option value="clip">剪辑特效（Clip Effect）</option>
  <option value="drop">降落特效（Drop Effect）</option>
  <option value="explode">爆炸特效（Explode Effect）</option>
  <option value="fold">折叠特效（Fold Effect）</option>
  <option value="highlight">突出特效（Highlight Effect）</option>
  <option value="puff">膨胀特效（Puff Effect）</option>
  <option value="pulsate">跳动特效（Pulsate Effect）</option>
  <option value="scale">缩放特效（Scale Effect）</option>
  <option value="shake">震动特效（Shake Effect）</option>
  <option value="size">尺寸特效（Size Effect）</option>
  <option value="slide">滑动特效（Slide Effect）</option>
</select>

<a href="#" id="button" class="ui-state-default ui-corner-all">运行

</body>
</html>
```

jQuery UI 实例 - 添加 Class（Add Class）

当动画样式改变时，为匹配的元素集合内的每个元素添加指定的 Class。

如需了解更多有关 `.addClass()` 方法的细节，请查看 API 文档 .addClass()。

.addClass() 演示

点击按钮预览特效。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 特效 - .addClass() 演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
    .toggler { width: 500px; height: 200px; position: relative; }
    #button { padding: .5em 1em; text-decoration: none; }
    #effect { width: 240px;  padding: 1em;  font-size: 1.2em; borde
    .newClass { text-indent: 40px; letter-spacing: .4em; width: 410
  </style>
  <script>
  $(function() {
    $( "#button" ).click(function() {
      $( "#effect" ).addClass( "newClass", 1000, callback );
      return false;
    });

    function callback() {
      setTimeout(function() {
        $( "#effect" ).removeClass( "newClass" );
      }, 1500 );
    }
  });
  </script>
</head>
<body>

<div class="toggler">
  <div id="effect" class="ui-corner-all">
      Etiam libero neque, luctus a, eleifend nec, semper at, lorem.
  </div>
</div>

<a href="#" id="button" class="ui-state-default ui-corner-all">运行

</body>
</html>
```

jQuery UI 实例 - 移除 Class（Remove Class）

当动画样式改变时，为匹配的元素集合内的每个元素移除指定的 Class。

如需了解更多有关 `.removeClass()` 方法的细节，请查看 API 文档 .removeClass()。

.removeClass() 演示

点击按钮预览特效。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 特效 - .removeClass() 演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  .toggler { width: 500px; height: 200px; position: relative; }
  #button { padding: .5em 1em; text-decoration: none; }
  #effect { position: relative;  width: 240px;  padding: 1em;  lett
  .newClass { text-indent: 40px; letter-spacing: .4em; width: 410px
  </style>
  <script>
  $(function() {
    $( "#button" ).click(function() {
      $( "#effect" ).removeClass( "newClass", 1000, callback );
      return false;
    });

    function callback() {
      setTimeout(function() {
        $( "#effect" ).addClass( "newClass" );
      }, 1500 );
    }
  });
  </script>
</head>
<body>

<div class="toggler">
  <div id="effect" class="newClass ui-corner-all">
    Etiam libero neque, luctus a, eleifend nec, semper at, lorem. S
  </div>
</div>

<a href="#" id="button" class="ui-state-default ui-corner-all">运行

</body>
</html>
```

jQuery UI 实例 - 切换 Class（Toggle Class）

当动画样式改变时，根据 Class 是否存在以及 switch 参数的值，为匹配的元素集合内的每个元素添加或移除一个或多个 Class。

如需了解更多有关 `.toggleClass()` 方法的细节，请查看 API 文档 .toggleClass()。

.toggleClass() 演示

点击按钮预览特效。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 特效 - .toggleClass() 演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  .toggler { width: 500px; height: 200px; position: relative; }
  #button { padding: .5em 1em; text-decoration: none; }
  #effect {position: relative;  width: 240px;  padding: 1em; letter
  .newClass { text-indent: 40px; letter-spacing: .4em; width: 410px
  </style>
  <script>
  $(function() {
    $( "#button" ).click(function() {
      $( "#effect" ).toggleClass( "newClass", 1000 );
      return false;
    });
  });
  </script>
</head>
<body>

<div class="toggler">
  <div id="effect" class="newClass ui-corner-all">
      Etiam libero neque, luctus a, eleifend nec, semper at, lorem.
  </div>
</div>

<a href="#" id="button" class="ui-state-default ui-corner-all">运行

</body>
</html>
```

jQuery UI 实例 - 转换 Class（Switch Class）

当动画样式改变时，为匹配的元素集合内的每个元素添加和移除指定的 Class。

如需了解更多有关 `.switchClass()` 方法的细节，请查看 API 文档 .switchClass()。

.switchClass() 演示

点击按钮预览特效。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 特效 - .switchClass() 演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  .toggler { width: 500px; height: 200px; position: relative; }
  #button { padding: .5em 1em; text-decoration: none; }
  #effect { position: relative; }
  .newClass { width: 240px;  padding: 1em; letter-spacing: 0; font-
  .anotherNewClass { text-indent: 40px; letter-spacing: .4em; width
  </style>
  <script>
  $(function() {
    $( "#button" ).click(function(){
      $( ".newClass" ).switchClass( "newClass", "anotherNewClass",
      $( ".anotherNewClass" ).switchClass( "anotherNewClass", "newC
      return false;
    });
  });
  </script>
</head>
<body>

<div class="toggler">
  <div id="effect" class="newClass ui-corner-all">
      Etiam libero neque, luctus a, eleifend nec, semper at, lorem.
  </div>
</div>
<a href="#" id="button" class="ui-state-default ui-corner-all">运行

</body>
</html>
```

jQuery UI 实例 - 颜色动画（Color Animation）

使用 .animate() 实现颜色动画效果。

如需了解更多有关颜色动画（Color Animation）的细节，请查看 API 文档 颜色动画（Color Animation）。

jQuery UI 捆绑了 jQuery Color 插件，jQuery Color 插件提供了颜色动画及其他许多与颜色相关的实用功能。

动画（Animation）演示

点击按钮预览特效。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 特效 - 动画（Animation）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
    .toggler { width: 500px; height: 200px; position: relative; }
    #button { padding: .5em 1em; text-decoration: none; }
    #effect { width: 240px; height: 135px; padding: 0.4em; positior
    #effect h3 { margin: 0; padding: 0.4em; text-align: center; }
  </style>
  <script>
  $(function() {
    var state = true;
    $( "#button" ).click(function() {
      if ( state ) {
        $( "#effect" ).animate({
          backgroundColor: "#aa0000",
          color: "#fff",
          width: 500
        }, 1000 );
      } else {
        $( "#effect" ).animate({
          backgroundColor: "#fff",
          color: "#000",
          width: 240
        }, 1000 );
      }
      state = !state;
    });
```

```
  });
  </script>
</head>
<body>

<div class="toggler">
  <div id="effect" class="ui-widget-content ui-corner-all">
    <h3 class="ui-widget-header ui-corner-all">动画（Animation）</h3
    <p>
      Etiam libero neque, luctus a, eleifend nec, semper at, lorem.
    </p>
  </div>
</div>

<a href="#" id="button" class="ui-state-default ui-corner-all">切换

</body>
</html>
```

jQuery UI 实例 - 定位（Position）

相对窗口、文档、锚、光标/鼠标等元素定位一个元素。

如需了解更多有关 `.position()` 方法的细节，请查看 API 文档 .position()。

默认功能

使用表单控件配置位置，或者拖拽被定位的元素来修改它的偏移量。向四周拖拽父元素来查看碰撞检测。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 定位（Position） - 默认功能</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #parent {
    width: 60%;
    height: 40px;
    margin: 10px auto;
    padding: 5px;
    border: 1px solid #777;
    background-color: #fbca93;
    text-align: center;
  }
  .positionable {
    position: absolute;
    display: block;
    right: 0;
    bottom: 0;
    background-color: #bcd5e6;
    text-align: center;
  }
  #positionable1 {
    width: 75px;
    height: 75px;
  }
  #positionable2 {
    width: 120px;
    height: 40px;
  }
  select, input {
    margin-left: 15px;
```

```
  }
  </style>
  <script>
  $(function() {
    function position() {
      $( ".positionable" ).position({
        of: $( "#parent" ),
        my: $( "#my_horizontal" ).val() + " " + $( "#my_vertical" ).
        at: $( "#at_horizontal" ).val() + " " + $( "#at_vertical" ).
        collision: $( "#collision_horizontal" ).val() + " " + $( "#
      });
    }

    $( ".positionable" ).css( "opacity", 0.5 );

    $( "select, input" ).bind( "click keyup change", position );

    $( "#parent" ).draggable({
      drag: position
    });

    position();
  });
  </script>
</head>
<body>

<div id="parent">
  <p>
  这是父元素的位置。
  </p>
</div>

<div class="positionable" id="positionable1">
  <p>
  to position
  </p>
</div>

<div class="positionable" id="positionable2">
  <p>
  to position 2
  </p>
</div>

<div style="padding: 20px; margin-top: 75px;">
  定位...
  <div style="padding-bottom: 20px;">
    <b>my:</b>
    <select id="my_horizontal">
      <option value="left">left</option>
      <option value="center">center</option>
      <option value="right">right</option>
```

```
      </select>
      <select id="my_vertical">
        <option value="top">top</option>
        <option value="center">center</option>
        <option value="bottom">bottom</option>
      </select>
    </div>
    <div style="padding-bottom: 20px;">
      <b>at:</b>
      <select id="at_horizontal">
        <option value="left">left</option>
        <option value="center">center</option>
        <option value="right">right</option>
      </select>
      <select id="at_vertical">
        <option value="top">top</option>
        <option value="center">center</option>
        <option value="bottom">bottom</option>
      </select>
    </div>
    <div style="padding-bottom: 20px;">
      <b>collision:</b>
      <select id="collision_horizontal">
        <option value="flip">flip</option>
        <option value="fit">fit</option>
        <option value="flipfit">flipfit</option>
        <option value="none">none</option>
      </select>
      <select id="collision_vertical">
        <option value="flip">flip</option>
        <option value="fit">fit</option>
        <option value="flipfit">flipfit</option>
        <option value="none">none</option>
      </select>
    </div>
  </div>

</body>
</html>
```

图像循环

一个照片浏览器的原型，使用 Position 分别把图片定为在左边、中间、右边，然后循环显示。使用顶部的链接来循环图像，或者在图像的左侧或右侧点击来循环图像。请注意，当调整窗口大小时，会重新定位图像。

```
<!doctype html>
<html lang="en">
<head>
```

```
<meta charset="utf-8">
<title>jQuery UI 定位（Position） - 图像循环</title>
<link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
<script src="//code.jquery.com/jquery-1.9.1.js"></script>
<script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
<link rel="stylesheet" href="http://jqueryui.com/resources/demos/
<style>
body {
  margin: 0;
}
#container {
  overflow: hidden;
  position: relative;
  height: 400px;
}

img {
  position: absolute;
}
</style>
<script>
$(function() {
  // 重构部件，去除这些插件方法
  $.fn.left = function( using ) {
    return this.position({
      my: "right middle",
      at: "left+25 middle",
      of: "#container",
      collision: "none",
      using: using
    });
  };
  $.fn.right = function( using ) {
    return this.position({
      my: "left middle",
      at: "right-25 middle",
      of: "#container",
      collision: "none",
      using: using
    });
  };
  $.fn.center = function( using ) {
    return this.position({
      my: "center middle",
      at: "center middle",
      of: "#container",
      using: using
    });
  };

  $( "img:eq(0)" ).left();
  $( "img:eq(1)" ).center();
  $( "img:eq(2)" ).right();
```

```
    function animate( to ) {
      $( this ).stop( true, false ).animate( to );
    }
    function next( event ) {
      event.preventDefault();
      $( "img:eq(2)" ).center( animate );
      $( "img:eq(1)" ).left( animate );
      $( "img:eq(0)" ).right().appendTo( "#container" );
    }
    function previous( event ) {
      event.preventDefault();
      $( "img:eq(0)" ).center( animate );
      $( "img:eq(1)" ).right( animate );
      $( "img:eq(2)" ).left().prependTo( "#container" );
    }
    $( "#previous" ).click( previous );
    $( "#next" ).click( next );

    $( "img" ).click(function( event ) {
      $( "img" ).index( this ) === 0 ? previous( event ) : next( ev
    });

    $( window ).resize(function() {
      $( "img:eq(0)" ).left( animate );
      $( "img:eq(1)" ).center( animate );
      $( "img:eq(2)" ).right( animate );
    });
  });
  </script>
</head>
<body>

<div id="container">
  <img src="img/earth.jpg" width="458" height="308" alt="earth">
  <img src="img/flight.jpg" width="512" height="307" alt="flight">
  <img src="img/rocket.jpg" width="300" height="353" alt="rocket">

  <a id="previous" href="#">上一个</a>
  <a id="next" href="#">下一个</a>
</div>

</body>
</html>
```

jQuery UI 实例 - 部件库（Widget Factory）

使用与所有 jQuery UI 小部件相同的抽象化来创建有状态的 jQuery 插件。

如需了解更多有关部件库（Widget Factory）的细节，请查看 API 文档 部件库（Widget Factory）。

默认功能

该演示展示了一个简单的使用部件库（jquery.ui.widget.js）创建的自定义的小部件。

三个区块是以不同的方式初始化的。点击改变他们的背景颜色。查看源码及注释理解工作原理。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 部件库（Widget Factory） - 默认功能</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  .custom-colorize {
    font-size: 20px;
    position: relative;
    width: 75px;
    height: 75px;
  }
  .custom-colorize-changer {
    font-size: 10px;
    position: absolute;
    right: 0;
    bottom: 0;
  }
  </style>
  <script>
  $(function() {
    // 部件定义，其中 "custom" 是命名空间，"colorize" 是部件名称
    $.widget( "custom.colorize", {
      // 默认选项
      options: {
        red: 255,
        green: 0,
        blue: 0,
```

```
        // 回调
        change: null,
        random: null
      },

      // 构造函数
      _create: function() {
        this.element
          // 添加一个主题化的 class
          .addClass( "custom-colorize" )
          // 防止双击来选择文本
          .disableSelection();

        this.changer = $( "<button>", {
          text: "改变",
          "class": "custom-colorize-changer"
        })
        .appendTo( this.element )
        .button();

        // 绑定 changer 按钮上的 click 事件到 random 方法
        this._on( this.changer, {
          // 当小部件被禁用时，_on 不会调用 random
          click: "random"
        });
        this._refresh();
      },

      // 当创建及之后改变选项时调用
      _refresh: function() {
        this.element.css( "background-color", "rgb(" +
          this.options.red +"," +
          this.options.green + "," +
          this.options.blue + ")"
        );

        // 触发一个回调/事件
        this._trigger( "change" );
      },

      // 一个改变颜色为随机值的公共方法
      // 可通过 .colorize( "random" ) 直接调用
      random: function( event ) {
        var colors = {
          red: Math.floor( Math.random() * 256 ),
          green: Math.floor( Math.random() * 256 ),
          blue: Math.floor( Math.random() * 256 )
        };

        // 触发一个事件，检查是否已取消
        if ( this._trigger( "random", event, colors ) !== false )
          this.option( colors );
        }
```

```
    },

    // 自动移除通过 _on 绑定的事件
    // 在这里重置其他的修改
    _destroy: function() {
      // 移除生成的元素
      this.changer.remove();

      this.element
        .removeClass( "custom-colorize" )
        .enableSelection()
        .css( "background-color", "transparent" );
    },

    // _setOptions 是通过一个带有所有改变的选项的哈希来调用的
    // 当改变选项时总是刷新
    _setOptions: function() {
      // _super 和 _superApply
      this._superApply( arguments );
      this._refresh();
    },

    // _setOption 是为每个独立的改变的选项调用的
    _setOption: function( key, value ) {
      // 防止无效的颜色值
      if ( /red|green|blue/.test(key) && (value < 0 || value > 25
        return;
      }
      this._super( key, value );
    }
  });

  // 通过默认选项进行初始化
  $( "#my-widget1" ).colorize();

  // 通过两个自定义的选项进行初始化
  $( "#my-widget2" ).colorize({
    red: 60,
    blue: 60
  });

  // 通过自定义的 green 值和一个只允许颜色足够绿的随机的回调进行初始化
  $( "#my-widget3" ).colorize( {
    green: 128,
    random: function( event, ui ) {
      return ui.green > 128;
    }
  });

  // 点击切换 enabled/disabled
  $( "#disable" ).click(function() {
    // 为每个小部件使用自定义的选择器来找到所有的实例
    // 所有的实例一起切换，所以我们可以从第一个开始检查状态
```

```
      if ( $( ":custom-colorize" ).colorize( "option", "disabled" )
        $( ":custom-colorize" ).colorize( "enable" );
      } else {
        $( ":custom-colorize" ).colorize( "disable" );
      }
    });

    // 在初始化后，点击设置选项
    $( "#black" ).click( function() {
      $( ":custom-colorize" ).colorize( "option", {
        red: 0,
        green: 0,
        blue: 0
      });
    });
  });
  </script>
</head>
<body>

<div>
  <div id="my-widget1">改变颜色</div>
  <div id="my-widget2">改变颜色</div>
  <div id="my-widget3">改变颜色</div>
  <button id="disable">切换 disabled 选项</button>
  <button id="black">改为黑色</button>
</div>

</body>
</html>
```

jQuery UI API 参考

jQuery UI API 类别 - 特效（Effects）

jQuery UI 在jQuery 内置的特效上添加了一些功能。jQuery UI 支持颜色动画和 Class 转换，同时也提供了一些额外的 Easings。另外，提供了一套完整的定制特效，供显示和隐藏元素时或者只是添加一些视觉显示时使用。

API	描述	也属于类别
.addClass()	当动画样式改变时，为匹配的元素集合内的每个元素添加指定的 Class。	特效核心（Effects Core） | 方法重载（Method Overrides）
百叶窗特效（Blind Effect）	百叶窗特效（Blind Effect）通过将元素包裹在一个容器内，采用"拉百叶窗"效果来隐藏或显示元素。	
反弹特效（Bounce Effect）	反弹特效（Bounce Effect）反弹一个元素。当与隐藏或显示一起使用时，最后一次或第一次反弹会呈现淡入/淡出效果。	
剪辑特效（Clip Effect）	剪辑特效（Clip Effect）通过垂直或水平方向夹剪元素来隐藏或显示一个元素。	
颜色动画（Color Animation）	使用 .animate() 实现颜色动画效果。	
降落特效（Drop Effect）	降落特效（Drop Effect）通过单个方向滑动的淡入淡出来隐藏或显示一个元素。	
Easings	Easing 函数指定动画在不同点上的行进速度。	
.effect()	对一个元素应用动画特效。	特效核心（Effects Core） | 方法（Method）
爆炸特效（Explode Effect）	爆炸特效（Explode Effect）通过把元素裂成碎片来隐藏或显示一个元素。	
淡入淡出特效（Fade Effect）	淡入淡出特效（Fade Effect）通过淡入淡出元素来隐藏或显示一个元素。	

（Fold Effect）	素来隐藏或显示一个元素。	
.hide()	使用自定义效果来隐藏匹配的元素。	特效核心（Effects Core） | 方法重载（Method Overrides） | 方法（Method）
突出特效（Highlight Effect）	突出特效（Highlight Effect）通过首先改变背景颜色来隐藏或显示一个元素。	
膨胀特效（Puff Effect）	通过在缩放元素的同时隐藏元素来创建膨胀特效（Puff Effect）。	
跳动特效（Pulsate Effect）	跳动特效（Pulsate Effect）通过跳动来隐藏或显示一个元素。	
.removeClass()	当动画样式改变时，为匹配的元素集合内的每个元素移除指定的 Class。	特效核心（Effects Core） | 方法重载（Method Overrides）
缩放特效（Scale Effect）	按照某个百分比缩放元素。	
震动特效（Shake Effect）	垂直或水平方向多次震动元素。	
.show()	使用自定义效果来显示匹配的元素。	特效核心（Effects Core） | 方法重载（Method Overrides） | 方法（Method）
尺寸特效（Size Effect）	调整元素尺寸到指定宽度和高度。	
滑动特效（Slide Effect）	把元素滑动出视区。	
.switchClass()	当动画样式改变时，为匹配的元素集合内的每个元素添加和移除指定的 Class。	特效核心（Effects Core）
.toggle()	使用自定义效果来显示或隐藏匹配的元素。	特效核心（Effects Core） | 方法重载（Method

	的元素。	Overrides） | 方法（Method）
.toggleClass()	当动画样式改变时，根据 Class 是否存在以及 switch 参数的值，为匹配的元素集合内的每个元素添加或移除一个或多个 Class。	特效核心（Effects Core） | 方法重载（Method Overrides）
转移特效（Transfer Effect）	把一个元素的轮廓转移到另一个元素。	

jQuery UI API - .addClass()

所属类别

特效（Effects） | 特效核心（Effects Core） | 方法重载（Method Overrides）

用法

描述：当动画样式改变时，为匹配的元素集合内的每个元素添加指定的 Class。

返回：jQuery

```
.addClass( className [, duration ] [, easing ] [, complete ] )
```

参数	类型	描述	默认值
className	String	要添加到每个匹配的元素的 class 属性中的一个或多个 class 名称，多个 class 名称用空格分隔。	
duration	Number 或 String	一个字符串或一个数字，指定动画将运行多久。	400
easing	String	一个字符串，指示要使用的 easing 函数。	swing
complete	Function()	一旦动画完成时要调用的函数。	

```
.addClass( className [, options ] )
```

参数	类型	描述
className	String	要添加到每个匹配的元素的 class 属性中的一个或多个 class 名称，多个 class 名称用空格分隔。
options	Object	所有的动画设置。所有的属性是可选的。

- **duration**（默认值：`400`） 类型：Number 或 String 描述：一个字符串或一个数字，指定动画将运行多久。
- **easing**（默认值：`swing`） 类型：String 描述：一个字符串，指示要使用的 easing 函数。
- **complete** 类型：Function() 描述：一旦动画完成时要调用的函数。

- **children**（默认值： `false` ） 类型：Boolean 描述：指定动画是否被应用到匹配元素的所有后代。此功能应慎重使用，因为判断元素是否是动画的后代的代价是很大的，会根据后代的数量线性增长。
- **queue**（默认值： `true` ） 类型：Boolean 或 String 描述：一个布尔值，指示是否将动画放在特效队列中。如果是 false，动画将立即开始。自 **jQuery 1.7** 起，queue 选项也接受一个字符串，在这种情况下，动画添加到由字符串表示的队列中。

与原生的 CSS 过渡相似，jQuery UI 的 class 动画提供了一个平稳的从一个状态转换到另一个状态的过渡，同时确保所有样式变化的细节是通过 CSS 来完成的，而不需要使用 JavaScript。所有的 class 动画方法，包括 `.addClass()` ，允许自定义动画持续时间和 easing 函数，在动画完成时也提供了一个回调。

并非所有的样式都可以进行动画添加。例如，对背景图像进行动画化。任何不能动画化的样式都将在动画的结尾改变。

该插件扩展自 jQuery 内置的 .addClass() 方法。如果 jQuery UI 未加载，调用 `.addClass()` 方法不会直接失败，因为该方法在 jQuery 中存在。但是不会发生预期的行为。

实例

添加 class "big-blue" 到匹配的元素。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>.addClass() 演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  div {
    width: 100px;
    height: 100px;
    background-color: #ccc;
  }
  .big-blue {
    width: 200px;
    height: 200px;
    background-color: #00f;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<div></div>

<script>
$( "div" ).click(function() {
  $( this ).addClass( "big-blue", 1000, "easeOutBounce" );
});
</script>

</body>
</html>
```

jQuery UI API - 百叶窗特效（Blind Effect）

所属类别

特效（Effects）

用法

描述：百叶窗特效（Blind Effect）通过将元素包裹在一个容器内，采用"拉百叶窗"效果来隐藏或显示元素。

```
blind
```

参数	类型	描述	默认值
direction	String	隐藏元素时拉动百叶窗的方向，显示元素时拉动百叶窗的方向。	

可能的值：up、down、left、right、vertical、horizontal。 | "up" |

容器应用 `overflow: hidden` 时，高度的变化会影响到元素的可见性。

实例

使用百叶窗特效（Blind Effect）切换一个 div。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>百叶窗特效（Blind Effect）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<p>点击任意地方进行切换。</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).toggle( "blind" );
});
</script>

</body>
</html>
```

jQuery UI API - 反弹特效（Bounce Effect）

所属类别

特效（Effects）

用法

描述：反弹特效（Bounce Effect）反弹一个元素。当与隐藏或显示一起使用时，最后一次或第一次反弹会呈现淡入/淡出效果。

```
bounce
```

参数	类型	描述	默认值
distance	Number	最大的反弹距离，以像素为单位。	20
times	Integer	元素反弹的次数。当隐藏或显示时，会为淡入/淡出效果添加半个反弹。	5

实例

使用反弹特效（Bounce Effect）切换一个 div。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>反弹特效（Bounce Effect）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<p>点击任意地方进行切换。</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).toggle( "bounce", { times: 3 }, "slow" );
});
</script>

</body>
</html>
```

jQuery UI API - 剪辑特效（Clip Effect）

所属类别

特效（Effects）

用法

描述：剪辑特效（Clip Effect）通过垂直或水平方向夹剪元素来隐藏或显示一个元素。

```
clip
```

参数	类型	描述	默认值
direction	String	剪辑特效隐藏或显示元素的方向。	

vertical 剪辑上下边缘，horizontal 剪辑左右边缘。 | "vertical" |

实例

使用剪辑特效（Clip Effect）切换一个 div。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>剪辑特效（Clip Effect）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<p>点击任意地方进行切换。</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).toggle( "clip" );
});
</script>

</body>
</html>
```

jQuery UI API - 颜色动画（Color Animation）

jQuery UI 特效核心添加了使用 `rgb()` 、 `rgba()` 、十六进制值或者颜色名比如 `"aqua"` 来动态改变 color 属性的功能。只需要包含 jQuery UI 特效核心文件， `.animate()` 就会支持颜色。

支持下列属性：

- `backgroundColor`
- `borderBottomColor`
- `borderLeftColor`
- `borderRightColor`
- `borderTopColor`
- `color`
- `columnRuleColor`
- `outlineColor`
- `textDecorationColor`
- `textEmphasisColor`

对颜色动画的支持来自 jQuery Color 插件。Color 插件提供了一些用于颜色的函数。如需查看完整文档，请访问 jQuery Color 文档。

Class 动画（Class Animations）

虽然可以直接对 color 属性进行动画化，但是通常采用另一种更好的方法，即在一个 class 中包含样式。jQuery UI 提供了一些动态添加或去除 CSS 类的方法，分别是 `.addClass()` 、 `.removeClass()` 、 `.toggleClass()` 和 `.switchClass()` 。这些方法将自动确定哪些属性需要改变，哪些需要应用适当的动画。

实例

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>颜色动画（Color Animation）演示</title>
  <link rel="stylesheet" href="http://code.jquery.com/ui/1.10.4/the
  <style>
  #elem {
    color: #006;
    background-color: #aaa;
    font-size: 25px;
    padding: 1em;
    text-align: center;
  }
  </style>
  <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
  <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></sc
</head>
<body>

<div id="elem">颜色动画</div>
<button id="toggle">改变颜色</button>

<script>
$( "#toggle" ).click(function() {
  $( "#elem" ).animate({
    color: "green",
    backgroundColor: "rgb( 20, 20, 20 )"
  });
});
</script>

</body>
</html>
```

jQuery UI API - 降落特效（Drop Effect）

所属类别

特效（Effects）

用法

描述：降落特效（Drop Effect）通过单个方向滑动的淡入淡出来隐藏或显示一个元素。

```
drop
```

参数	类型	描述	默认值
direction	String	隐藏元素时元素降落的方向，显示元素时元素出现的方向。	

可能的值：up、down、left、right。 | "left" |

实例

使用降落特效（Blind Effect）切换一个 div。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>降落特效（Drop Effect）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<p>点击任意地方进行切换。</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).toggle( "drop" );
});
</script>

</body>
</html>
```

jQuery UI API - Easings

Easing 函数指定动画在不同点上的行进速度。jQuery 核心带有两个 easings：一个是 `linear`，整个动画以一个不变的速度行进；另一个是 `swing`（jQuery 核心默认的 easing），行进速度在动画开始和结束时比在动画中间时稍慢。jQuery UI 提供了一些额外的 easing 函数，范围从摆动行为上的变化到定制特效，比如弹跳。

一些 easings 会在动画中产生负值。根据动画的不同属性，实际值可能为零。例如，您可以把 `left` 取为负值，但是不能把 `height` 或 `opacity` 取为负值。

想要更好地理解一个 easing 如何影响一个动画，需多花时间研究方程图。请看下面所列出的 jQuery UI 中所有可用动画的曲线图。





jQuery UI API - .effect()

所属类别

特效（Effects） | 特效核心（Effects Core） | 方法（Methods）

用法

描述：对一个元素应用动画特效。

返回：jQuery

```
.effect( effect [, options ] [, duration ] [, complete ] )
```

参数	类型	描述	默认值
effect	String	一个字符串，指示要使用哪一种特效。	
options	Object	特效具体的设置和 easing。	
duration	Number 或 String	一个字符串或一个数字，指定动画将运行多久。	400
complete	Function()	一旦动画完成时要调用的函数。	

```
.effect( options )
```

参数	类型	描述
options	Object	所有的动画设置。唯一一个必需的属性是 effect。

- **effect** 类型：String 描述：一个字符串，指示要使用哪一种特效。
- **easing**（默认值：`"swing"`） 类型：String 描述：一个字符串，指示要使用的 easing 函数。
- **duration**（默认值：`400`） 类型：Number 或 String 描述：一个字符串或一个数字，指定动画将运行多久。
- **complete** 类型：Function() 描述：一旦动画完成时要调用的函数。
- **queue**（默认值：`true`） 类型：Boolean 或 String 描述：一个布尔值，指示是否将动画放在特效队列中。如果是 false，动画将立即开始。自 **jQuery 1.7** 起，queue 选项也接受一个字符串，在这种情况下，动画添加到由字符串表示的队列中。

`.effect()` 方法对一个元素应用了一个命名的动画 特效 。许多特效也支持显示或隐藏模式，这些可通过 `.show()` 、 `.hide()` 和 `.toggle()` 方法来完成。

实例

对一个 div 应用反弹特效（Bounce Effect）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>.effect() 演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  div {
    width: 100px;
    height: 100px;
    background: #ccc;
    border: 1px solid #000;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<p>点击任意地方应用特效。</p>
<div></div>

<script>
$( document ).click(function() {
  $( "div" ).effect( "bounce", "slow" );
});
</script>

</body>
</html>
```

jQuery UI API - 爆炸特效（Explode Effect）

所属类别

特效（Effects）

用法

描述：爆炸特效（Explode Effect）通过把元素裂成碎片来隐藏或显示一个元素。

```
explode
```

参数	类型	描述	默认值
pieces	Integer	爆炸裂开的碎片数目，应该是个平方数，任何其他值被舍入到最近的平方数。	9

实例

使用爆炸特效（Explode Effect）切换一个 div。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>爆炸特效（Explode Effect）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<p>点击任意地方进行切换。</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).toggle( "explode" );
});
</script>

</body>
</html>
```

jQuery UI API - 淡入淡出特效（Fade Effect）

所属类别

特效（Effects）

用法

描述：淡入淡出特效（Fade Effect）通过淡入淡出元素来隐藏或显示一个元素。

```
fade
```

实例

使用淡入淡出特效（Fade Effect）切换一个 div。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>淡入淡出特效（Fade Effect）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<p>点击任意地方进行切换。</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).toggle( "fade" );
});
</script>

</body>
</html>
```

jQuery UI API - 折叠特效（Fold Effect）

所属类别

特效（Effects）

用法

描述：折叠特效（Fold Effect）通过折叠元素来隐藏或显示一个元素。

```
fold
```

参数	类型	描述	默认值
size	Number 或 String	被折叠元素的尺寸。	15
horizFirst	Boolean	当折叠时是否先进行水平方向的折叠。请记得，显示的时候与隐藏的时候顺序相反。	false

实例

使用折叠特效（Fold Effect）切换一个 div。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>折叠特效（Fold Effect）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<p>点击任意地方进行切换。</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).toggle( "fold" );
});
</script>

</body>
</html>
```

jQuery UI API - .hide()

所属类别

特效（Effects） | 特效核心（Effects Core） | 方法重载（Method Overrides） | 方法（Methods）

用法

描述：使用自定义效果来隐藏匹配的元素。

返回：jQuery

```
.hide( effect [, options ] [, duration ] [, complete ] )
```

参数	类型	描述	默认值
effect	String	一个字符串，指示要使用哪一种特效。	
options	Object	特效具体的设置和 easing。	
duration	Number 或 String	一个字符串或一个数字，指定动画将运行多久。	400
complete	Function()	一旦动画完成时要调用的函数。	

```
.hide( options )
```

参数	类型	描述
options	Object	所有的动画设置。唯一一个必需的属性是 effect。

- **effect** 类型：String 描述：一个字符串，指示要使用哪一种特效。
- **easing**（默认值：`"swing"`） 类型：String 描述：一个字符串，指示要使用的 easing 函数。
- **duration**（默认值：`400`） 类型：Number 或 String 描述：一个字符串或一个数字，指定动画将运行多久。
- **complete** 类型：Function() 描述：一旦动画完成时要调用的函数。
- **queue**（默认值：`true`） 类型：Boolean 或 String 描述：一个布尔值，指示是否将动画放在特效队列中。如果是 false，动画将立即开始。自 **jQuery 1.7** 起，queue 选项也接受一个字符串，在这种情况下，动画添加到由字符串

表示的队列中。

该插件扩展自 jQuery 内置的 .hide() 方法。如果 jQuery UI 未加载，调用 `.hide()` 方法不会直接失败，因为该方法在 jQuery 中存在。但是不会发生预期的行为。

实例

使用降落特效（Drop Effect）隐藏一个 div。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>.hide() 演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  div {
    width: 100px;
    height: 100px;
    background: #ccc;
    border: 1px solid #000;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<button>隐藏 div</button>
<div></div>

<script>
$( "button" ).click(function() {
  $( "div" ).hide( "drop", { direction: "down" }, "slow" );
});
</script>

</body>
</html>
```

jQuery UI API - 突出特效（Highlight Effect）

所属类别

特效（Effects）

用法

描述：突出特效（Highlight Effect）通过首先改变背景颜色来隐藏或显示一个元素。

```
highlight
```

参数	类型	描述	默认值
color	String	动画期间使用的背景颜色。	"#ffff99"

实例

使用突出特效（Highlight Effect）切换一个 div。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>突出特效（Highlight Effect）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<p>点击任意地方进行切换。</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).toggle( "highlight" );
});
</script>

</body>
</html>
```

jQuery UI API - 膨胀特效（Puff Effect）

所属类别

特效（Effects）

用法

描述：通过在缩放元素的同时隐藏元素来创建膨胀特效（Puff Effect）。

```
puff
```

参数	类型	描述	默认值
percent	Number	要缩放到的百分比。	150

实例

使用膨胀特效（Puff Effect）切换一个 div。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>膨胀特效（Puff Effect）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<p>点击任意地方进行切换。</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).toggle( "puff" );
});
</script>

</body>
</html>
```

jQuery UI API - 跳动特效（Pulsate Effect）

所属类别

特效（Effects）

用法

描述：跳动特效（Pulsate Effect）通过跳动来隐藏或显示一个元素。

```
pulsate
```

参数	类型	描述	默认值
times	Integer	元素跳动的次数。当隐藏或显示时，会添加半个跳动。	5

实例

使用跳动特效（Pulsate Effect）切换一个 div。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>跳动特效（Pulsate Effect）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<p>点击任意地方进行切换。</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).toggle( "pulsate" );
});
</script>

</body>
</html>
```

jQuery UI API - .removeClass()

所属类别

特效（Effects） | 特效核心（Effects Core） | 方法重载（Method Overrides）

用法

描述：当动画样式改变时，为匹配的元素集合内的每个元素移除指定的 Class。

返回：jQuery

```
.removeClass( className [, duration ] [, easing ] [, complete ] )
```

参数	类型	描述	默认值
className	String	要从每个匹配的元素的 class 属性中移除的一个或多个 class 名称，多个 class 名称用空格分隔。	
duration	Number 或 String	一个字符串或一个数字，指定动画将运行多久。	400
easing	String	一个字符串，指示要使用的 easing 函数。	swing
complete	Function()	一旦动画完成时要调用的函数。	

```
.removeClass( className [, options ] )
```

参数	类型	描述
className	String	要从每个匹配的元素的 class 属性中移除的一个或多个 class 名称，多个 class 名称用空格分隔。
options	Object	所有的动画设置。所有的属性是可选的。

- **duration**（默认值：`400`） 类型：Number 或 String 描述：一个字符串或一个数字，指定动画将运行多久。
- **easing**（默认值：`swing`） 类型：String 描述：一个字符串，指示要使用的 easing 函数。

- **complete** 类型：Function() 描述：一旦动画完成时要调用的函数。
- **children**（默认值： `false` ） 类型：Boolean 描述：指定动画是否被应用到匹配元素的所有后代。此功能应慎重使用，因为判断元素是否是动画的后代的代价是很大的，会根据后代的数量线性增长。
- **queue**（默认值： `true` ） 类型：Boolean 或 String 描述：一个布尔值，指示是否将动画放在特效队列中。如果是 false，动画将立即开始。自 **jQuery 1.7** 起，queue 选项也接受一个字符串，在这种情况下，动画添加到由字符串表示的队列中。

与原生的 CSS 过渡相似，jQuery UI 的 class 动画提供了一个平稳的从一个状态转换到另一个状态的过渡，同时确保所有样式变化的细节是通过 CSS 来完成的，而不需要使用 JavaScript。所有的 class 动画方法，包括 `.removeClass()` ，允许自定义动画持续时间和 easing 函数，在动画完成时也提供了一个回调。

并非所有的样式都可以进行动画添加。例如，对背景图像进行动画化。任何不能动画化的样式都将在动画的结尾改变。

该插件扩展了 jQuery 内置的 .removeClass() 方法。如果 jQuery UI 未加载，调用 `.removeClass()` 方法不会直接失败，因为该方法在 jQuery 中存在。但是不会发生预期的行为。

实例

从匹配的元素中移除 class "big-blue" 。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>.removeClass() 演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  div {
    width: 100px;
    height: 100px;
    background-color: #ccc;
  }
  .big-blue {
    width: 200px;
    height: 200px;
    background-color: #00f;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<div class="big-blue"></div>

<script>
$( "div" ).click(function() {
  $( this ).removeClass( "big-blue", 1000, "easeInBack" );
});
</script>

</body>
</html>
```

jQuery UI API - 缩放特效（Scale Effect）

所属类别

特效（Effects）

用法

描述：按照某个百分比缩放元素。

```
scale
```

参数	类型	描述	默认值
direction	String	特效的方向。可能的值："both"、"vertical" 或 "horizontal"。	"both"
origin	Array	消失点。	["middle", "center"]
percent	Number	要缩放到的百分比。	
scale	String	元素的哪个区域将被调整尺寸："both"、"box"、"content"。当值为 "box" 时，调整元素的边框（border）和内边距（padding）的尺寸。当值为 "content" 时，调整元素内的所有内容的尺寸。	"both"

实例

实例 **1**：

使用缩放特效（Scale Effect）切换一个 div。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>缩放特效（Scale Effect）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<p>点击任意地方进行切换。</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).toggle( "scale" );
});
</script>

</body>
</html>
```

实例 **2**：

只在一个方向上使用缩放特效（Scale Effect）切换一个 div。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>>缩放特效（Scale Effect）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<p>点击任意地方进行切换。</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).toggle({ effect: "scale", direction: "horizontal"
});
</script>

</body>
</html>
```

jQuery UI API - 震动特效（Shake Effect）

所属类别

特效（Effects）

用法

描述：垂直或水平方向多次震动元素。

```
shake
```

参数	类型	描述	默认值
direction	String	"left" 或 "right" 的值将水平震动元素，"up" 或 "down" 的值将垂直震动元素。该值指定元素沿轴线移动时第一步的方向。	"left"
distance	Number	要震动的距离。	20
times	Integer	要震动的次数。	3

实例

震动一个 div。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>震动特效（Shake Effect）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<p>点击任意地方进行震动。</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).effect( "shake" );
});
</script>

</body>
</html>
```

jQuery UI API - .show()

所属类别

特效（Effects） | 特效核心（Effects Core） | 方法重载（Method Overrides） | 方法（Methods）

用法

描述：使用自定义效果来显示匹配的元素。

返回：jQuery

```
.show( effect [, options ] [, duration ] [, complete ] )
```

参数	类型	描述	默认值
effect	String	一个字符串，指示要使用哪一种特效。	
options	Object	特效具体的设置和 easing。	
duration	Number 或 String	一个字符串或一个数字，指定动画将运行多久。	400
complete	Function()	一旦动画完成时要调用的函数。	

```
.show( options )
```

参数	类型	描述
options	Object	所有的动画设置。唯一一个必需的属性是 effect。

- **effect** 类型：String 描述：一个字符串，指示要使用哪一种特效。
- **easing**（默认值：`"swing"`） 类型：String 描述：一个字符串，指示要使用的 easing 函数。
- **duration**（默认值：`400`） 类型：Number 或 String 描述：一个字符串或一个数字，指定动画将运行多久。
- **complete** 类型：Function() 描述：一旦动画完成时要调用的函数。
- **queue**（默认值：`true`） 类型：Boolean 或 String 描述：一个布尔值，指示是否将动画放在特效队列中。如果是 false，动画将立即开始。自 **jQuery 1.7** 起，queue 选项也接受一个字符串，在这种情况下，动画添加到由字符串

表示的队列中。

该插件扩展自 jQuery 内置的 .show() 方法。如果 jQuery UI 未加载，调用 `.show()` 方法不会直接失败，因为该方法在 jQuery 中存在。但是不会发生预期的行为。

实例

使用折叠特效（Fold Effect）显示一个 div。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>.show() 演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  div {
    display: none;
    width: 100px;
    height: 100px;
    background: #ccc;
    border: 1px solid #000;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<button>显示 div</button>
<div></div>

<script>
$( "button" ).click(function() {
  $( "div" ).show( "fold", 1000 );
});
</script>

</body>
</html>
```

jQuery UI API - 尺寸特效（Size Effect）

所属类别

特效（Effects）

用法

描述：调整元素尺寸到指定宽度和高度。

```
size
```

参数	类型	描述	默认值
to	Object	要调整到的高度和宽度。	
origin	Array	消失点。	["top", "left"]
scale	String	元素的哪个区域将被调整尺寸："both"、"box"、"content"。当值为 "box" 时，调整元素的边框（border）和内边距（padding）的尺寸。当值为 "content" 时，调整元素内的所有内容的尺寸。	"both"

实例

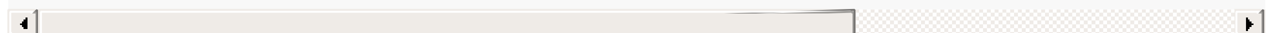
使用尺寸特效（Size Effect）调整元素尺寸。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>尺寸特效（Size Effect）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<p>点击任意地方进行尺寸调整。</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).effect( "size", {
    to: { width: 200, height: 60 }
  }, 1000 );
});
</script>

</body>
</html>
```



jQuery UI API - 滑动特效（Slide Effect）

所属类别

特效（Effects）

用法

描述：把元素滑动出视区。

```
slide
```

参数	类型	描述	默认值
direction	String	特效的方向。可能的值："left"、"right"、"up"、"down"。	"both"
distance	Number	特效的距离。默认为元素的高度（height）还是宽度（width）取决于 direction 参数。可以设置为小于元素的宽度（width）/高度（height）的任意整数。	元素的 outerWidth

实例

使用滑动特效（Slide Effect）切换一个 div。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>滑动特效（Slide Effect）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<p>点击任意地方进行切换。</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).toggle( "slide" );
});
</script>

</body>
</html>
```

jQuery UI API - .switchClass()

所属类别

特效（Effects） | 特效核心（Effects Core） | 方法重载（Method Overrides）

用法

描述：当动画样式改变时，为匹配的元素集合内的每个元素添加和移除指定的 Class。

返回：jQuery

```
.switchClass( removeClassName, addClassName [, duration ] [, easing
```

参数	类型	描述	默认值
removeClassName	String	要从每个匹配的元素的 class 属性中移除的一个或多个 class 名称，多个 class 名称用空格分隔。	
addClassName	String	要添加到每个匹配的元素的 class 属性中的一个或多个 class 名称，多个 class 名称用空格分隔。	
duration	Number 或 String	一个字符串或一个数字，指定动画将运行多久。	400
easing	String	一个字符串，指示要使用的 easing 函数。	swing
complete	Function()	一旦动画完成时要调用的函数。	

```
.switchClass( removeClassName, addClassName [, options ] )
```

参数	类型	描述
removeClassName	String	要从每个匹配的元素的 class 属性中移除的一个或多个 class 名称，多个 class 名称用空格分隔 。
addClassName	String	要添加到每个匹配的元素的 class 属性中的一个或多个 class 名称，多个 class 名称用空格分隔。
options	Object	所有的动画设置。所有的属性是可选的。

- **duration**（默认值： `400` ） 类型：Number 或 String 描述：一个字符串或一个数字，指定动画将运行多久。
- **easing**（默认值： `swing` ） 类型：String 描述：一个字符串，指示要使用的 easing 函数。
- **complete** 类型：Function() 描述：一旦动画完成时要调用的函数。
- **children**（默认值： `false` ） 类型：Boolean 描述：指定动画是否被应用到匹配元素的所有后代。此功能应慎重使用，因为判断元素是否是动画的后代的代价是很大的，会根据后代的数量线性增长。
- **queue**（默认值： `true` ） 类型：Boolean 或 String 描述：一个布尔值，指示是否将动画放在特效队列中。如果是 false，动画将立即开始。自 **jQuery 1.7** 起，queue 选项也接受一个字符串，在这种情况下，动画添加到由字符串表示的队列中。

`.switchClass()` 方法允许您在动画过渡中，添加 Class 的同时移除 Class。

与原生的 CSS 过渡相似，jQuery UI 的 class 动画提供了一个平稳的从一个状态转换到另一个状态的过渡，同时确保所有样式变化的细节是通过 CSS 来完成的，而不需要使用 JavaScript。所有的 class 动画方法，包括 `.switchClass()` ，允许自定义动画持续时间和 easing 函数，在动画完成时也提供了一个回调。

并非所有的样式都可以进行动画添加。例如，对背景图像进行动画化。任何不能动画化的样式都将在动画结束时改变。

该插件扩展自 jQuery 内置的 .switchClass() 方法。如果 jQuery UI 未加载，调用 `.switchClass()` 方法不会直接失败，因为该方法在 jQuery 中存在。但是不会发生预期的行为。

实例

添加 class "blue" 到匹配的元素，并从匹配的元素中移除 class "big"。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>.switchClass() 演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  div {
    width: 100px;
    height: 100px;
    background-color: #ccc;
  }
  .big {
    width: 200px;
    height: 200px;
  }
  .blue {
    background-color: #00f;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<div class="big"></div>

<script>
$( "div" ).click(function() {
  $( this ).switchClass( "big", "blue", 1000, "easeInOutQuad" );
});
</script>

</body>
</html>
```

jQuery UI API - .toggle()

所属类别

特效（Effects） | 特效核心（Effects Core） | 方法重载（Method Overrides） | 方法（Methods）

用法

描述：使用自定义效果来显示或隐藏匹配的元素。

返回：jQuery

```
.toggle( effect [, options ] [, duration ] [, complete ] )
```

参数	类型	描述	默认值
effect	String	一个字符串，指示要使用哪一种特效。	
options	Object	特效具体的设置和 easing。	
duration	Number 或 String	一个字符串或一个数字，指定动画将运行多久。	400
complete	Function()	一旦动画完成时要调用的函数。	

```
.toggle( options )
```

参数	类型	描述
options	Object	所有的动画设置。唯一一个必需的属性是 effect。

- **effect** 类型：String 描述：一个字符串，指示要使用哪一种特效。
- **easing**（默认值：`"swing"`） 类型：String 描述：一个字符串，指示要使用的 easing 函数。
- **duration**（默认值：`400`） 类型：Number 或 String 描述：一个字符串或一个数字，指定动画将运行多久。
- **complete** 类型：Function() 描述：一旦动画完成时要调用的函数。
- **queue**（默认值：`true`） 类型：Boolean 或 String 描述：一个布尔值，指示是否将动画放在特效队列中。如果是 false，动画将立即开始。自 **jQuery 1.7** 起，queue 选项也接受一个字符串，在这种情况下，动画添加到由字符串

表示的队列中。

该插件扩展自 jQuery 内置的 .toggle() 方法。如果 jQuery UI 未加载，调用 `.toggle()` 方法不会直接失败，因为该方法在 jQuery 中存在。但是不会发生预期的行为。

实例

使用折叠特效（Fold Effect）切换一个 div。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>.toggle() 演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  div {
    width: 100px;
    height: 100px;
    background: #ccc;
    border: 1px solid #000;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<button>切换 div</button>
<div></div>

<script>
$( "button" ).click(function() {
  $( "div" ).toggle( "fold", 1000 );
});
</script>

</body>
</html>
```

jQuery UI API - .toggleClass()

所属类别

特效（Effects） | 特效核心（Effects Core） | 方法重载（Method Overrides）

用法

描述：当动画样式改变时，根据 Class 是否存在以及 switch 参数的值，为匹配的元素集合内的每个元素添加或移除一个或多个 Class。

返回：jQuery

```
.toggleClass( className [, switch ] [, duration ] [, easing ] [, co
```

参数	类型	描述	默认值
ClassName	String	为匹配的元素集合中的每个元素要切换的一个或多个 class 名称，多个 class 名称用空格分隔 。	
switch	Boolean	一个布尔值，指定 class 应被添加还是被移除。	
duration	Number 或 String	一个字符串或一个数字，指定动画将运行多久。	400
easing	String	一个字符串，指示要使用的 easing 函数。	swing
complete	Function()	一旦动画完成时要调用的函数。	

```
.toggleClass( className [, switch ] [, options ] )
```

参数	类型	描述
ClassName	String	为匹配的元素集合中的每个元素要切换的一个或多个 class 名称，多个 class 名称用空格分隔 。
switch	Boolean	一个布尔值，指定 class 应被添加还是被移除。
options	Object	所有的动画设置。所有的属性是可选的。

- **duration**（默认值： `400` ） 类型：Number 或 String 描述：一个字符串或一个数字，指定动画将运行多久。
- **easing**（默认值： `swing` ） 类型：String 描述：一个字符串，指示要使用的 easing 函数。
- **complete** 类型：Function() 描述：一旦动画完成时要调用的函数。
- **children**（默认值： `false` ） 类型：Boolean 描述：指定动画是否被应用到匹配元素的所有后代。此功能应慎重使用，因为判断元素是否是动画的后代的代价是很大的，会根据后代的数量线性增长。
- **queue**（默认值： `true` ） 类型：Boolean 或 String 描述：一个布尔值，指示是否将动画放在特效队列中。如果是 false，动画将立即开始。自 **jQuery 1.7** 起，queue 选项也接受一个字符串，在这种情况下，动画添加到由字符串表示的队列中。

与原生的 CSS 过渡相似，jQuery UI 的 class 动画提供了一个平稳的从一个状态转换到另一个状态的过渡，同时确保所有样式变化的细节是通过 CSS 来完成的，而不需要使用 JavaScript。所有的 class 动画方法，包括 `.toggleClass()` ，允许自定义动画持续时间和 easing 函数，在动画完成时也提供了一个回调。

并非所有的样式都可以进行动画添加。例如，对背景图像进行动画化。任何不能动画化的样式都将在动画结束时改变。

该插件扩展自 jQuery 内置的 .toggleClass() 方法。如果 jQuery UI 未加载，调用 `.toggleClass()` 方法不会直接失败，因为该方法在 jQuery 中存在。但是不会发生预期的行为。

实例

为匹配的元素切换 class "big-blue"。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>.toggleClass() 演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  div {
    width: 100px;
    height: 100px;
    background-color: #ccc;
  }
  .big-blue {
    width: 200px;
    height: 200px;
    background-color: #00f;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<div></div>

<script>
$( "div" ).click(function() {
  $( this ).toggleClass( "big-blue", 1000, "easeOutSine" );
});
</script>

</body>
</html>
```

jQuery UI API - 转移特效（Transfer Effect）

所属类别

特效（Effects）

用法

描述：把一个元素的轮廓转移到另一个元素。

```
transfer
```

参数	类型	描述
className	String	转移的元素将接收的参数化的 class 名称。
to	String	jQuery 选择器，要转移到的元素。

当尝试两个元素之间的可视化交互时非常有用。

转移的元素本身带有 class `ui-effects-transfer`，其他的样式需要由您来定义，比如添加背景或边框。

实例

在绿色元素上点击把它转移到另一个元素。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>转移特效（Transfer Effect）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  div.green {
    width: 100px;
    height: 80px;
    background: green;
    border: 1px solid black;
    position: relative;
  }
  div.red {
    margin-top: 10px;
    width: 50px;
    height: 30px;
    background: red;
    border: 1px solid black;
    position: relative;
  }
  .ui-effects-transfer {
    border: 1px dotted black;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<div class="green"></div>
<div class="red"></div>

<script>
$( "div" ).click(function() {
  var i = 1 - $( "div" ).index( this );
  $( this ).effect( "transfer", { to: $( "div" ).eq( i ) }, 1000 ),
});
</script>

</body>
</html>
```

jQuery UI API 类别 - 特效核心（Effects Core）

由 jquery.ui.effect.js 提供的功能。除了下面列出的方法，jquery.ui.effect.js 还包括一些 Easings。

API	描述	也属于类别
.addClass()	当动画样式改变时，为匹配的元素集合内的每个元素添加指定的 Class。	特效（Effects） | 方法重载（Method Overrides）
颜色动画（Color Animation）	使用 .animate() 实现颜色动画效果。	
.effect()	对一个元素应用动画特效。	特效（Effects） | 方法（Method）
.hide()	使用自定义效果来隐藏匹配的元素。	特效（Effects） | 方法重载（Method Overrides） | 方法（Method）
.removeClass()	当动画样式改变时，为匹配的元素集合内的每个元素移除指定的 Class。	特效（Effects） | 方法重载（Method Overrides）
.show()	使用自定义效果来显示匹配的元素。	特效（Effects） | 方法重载（Method Overrides） | 方法（Method）
.switchClass()	当动画样式改变时，为匹配的元素集合内的每个元素添加和移除指定的 Class。	特效（Effects）
.toggle()	使用自定义效果来显示或隐藏匹配的元素。	特效（Effects） | 方法重载（Method Overrides） | 方法（Method）
.toggleClass()	当动画样式改变时，根据 Class 是否存在以及 switch 参数的值，为匹配的元素集合内的每个元素添加或移除一个或多个 Class。	特效（Effects） | 方法重载（Method Overrides）

jQuery UI API - 颜色动画（Color Animation）

jQuery UI 特效核心添加了使用 `rgb()` 、 `rgba()` 、十六进制值或者颜色名比如 `"aqua"` 来动态改变 color 属性的功能。只需要包含 jQuery UI 特效核心文件， `.animate()` 就会支持颜色。

支持下列属性：

- `backgroundColor`
- `borderBottomColor`
- `borderLeftColor`
- `borderRightColor`
- `borderTopColor`
- `color`
- `columnRuleColor`
- `outlineColor`
- `textDecorationColor`
- `textEmphasisColor`

对颜色动画的支持来自 jQuery Color 插件。Color 插件提供了一些用于颜色的函数。如需查看完整文档，请访问 jQuery Color 文档。

Class 动画（Class Animations）

虽然可以直接对 color 属性进行动画化，但是通常采用另一种更好的方法，即在一个 class 中包含样式。jQuery UI 提供了一些动态添加或去除 CSS 类的方法，分别是 `.addClass()` 、 `.removeClass()` 、 `.toggleClass()` 和 `.switchClass()` 。这些方法将自动确定哪些属性需要改变，哪些需要应用适当的动画。

实例

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>颜色动画（Color Animation）演示</title>
  <link rel="stylesheet" href="http://code.jquery.com/ui/1.10.4/the
  <style>
  #elem {
    color: #006;
    background-color: #aaa;
    font-size: 25px;
    padding: 1em;
    text-align: center;
  }
  </style>
  <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
  <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></sc
</head>
<body>

<div id="elem">颜色动画</div>
<button id="toggle">改变颜色</button>

<script>
$( "#toggle" ).click(function() {
  $( "#elem" ).animate({
    color: "green",
    backgroundColor: "rgb( 20, 20, 20 )"
  });
});
</script>

</body>
</html>
```

jQuery UI API 类别 - 交互（Interactions）

jQuery UI 提供了一套基于鼠标的交互。

API	描述	也属于类别
可拖拽小部件（Draggable Widget）	允许使用鼠标移动元素。	
可放置小部件（Droppable Widget）	为可拖拽小部件创建目标。	
鼠标交互（Mouse Interaction）	基本交互层。	实用工具（Utilities）
可调整尺寸小部件（Resizable Widget）	使用鼠标改变元素的尺寸。	
可选择小部件（Selectable Widget）	使用鼠标选择单个元素或一组元素。	
可排序小部件（Sortable Widget）	使用鼠标调整列表中或者网格中元素的排序。	

jQuery UI API - 可拖拽小部件（Draggable Widget）

所属类别

交互（Interactions）

用法

描述：允许使用鼠标移动元素。

版本新增：1.0

依赖：

- UI 核心（UI Core）
- 部件库（Widget Factory）
- 鼠标交互（Mouse Interaction）

注释：让被选元素可被鼠标拖拽。如果您不只是拖拽，而是拖拽 & 放置，请查看 jQuery UI 可放置（Droppable）插件，为可拖拽元素提供了一个放置目标。

快速导航

addClassesappendToaxiscancelconnectToSortablecontainmentcursorcursorAtde

addClasses

类型：Boolean

描述：如果设置为 `false`，将阻止 `ui-draggable` class 被添加。当在数百个元素上调用 `.draggable()` 时，这么设置有利于性能优化。

代码实例：

初始化带有指定 `addClasses` 选项的 draggable：

```
$( ".selector" ).draggable({ addClasses: false });
```

在初始化后，获取或设置 `addClasses` 选项：

```
// getter
var addClasses = $( ".selector" ).draggable( "option", "addClasses'

// setter
$( ".selector" ).draggable( "option", "addClasses", false );
```

默认值：true

appendTo

类型：jQuery 或 Element 或 Selector 或 String

描述：当拖拽时，draggable 助手（helper）要追加到哪一个元素。

支持多个类型：

- **jQuery**：一个 jQuery 对象，包含助手（helper）要追加到的元素。
- **Element**：要追加助手（helper）的元素。
- **Selector**：一个选择器，指定哪一个元素要追加助手（helper）。
- **String**：字符串 `"parent"` 将促使助手（helper）成为 draggable 的同级。

代码实例：

初始化带有指定 `appendTo` 选项的 draggable：

```
$( ".selector" ).draggable({ appendTo: "body" });
```

在初始化后，获取或设置 `appendTo` 选项：

```
// getter
var appendTo = $( ".selector" ).draggable( "option", "appendTo" );

// setter
$( ".selector" ).draggable( "option", "appendTo", "body" );
```

默认值："parent"

axis

类型：String

描述：约束在水平轴 (x) 或垂直轴 (y) 上拖拽。可能的值：`"x"`，`"y"`。

代码实例：

初始化带有指定 `axis` 选项的 draggable：

```
$( ".selector" ).draggable({ axis: "x" });
```

在初始化后，获取或设置 `axis` 选项：

```
// getter
var axis = $( ".selector" ).draggable( "option", "axis" );

// setter
$( ".selector" ).draggable( "option", "axis", "x" );
```

默认值：false

cancel

类型：Selector

描述：防止从指定的元素上开始拖拽。

代码实例：

初始化带有指定 `cancel` 选项的 draggable：

```
$( ".selector" ).draggable({ cancel: ".title" });
```

在初始化后，获取或设置 `cancel` 选项：

```
// getter
var cancel = $( ".selector" ).draggable( "option", "cancel" );

// setter
$( ".selector" ).draggable( "option", "cancel", ".title" );
```

默认值："input, textarea, button, select, option"

connectToSortable

类型：Selector

描述：允许 draggable 放置在指定的 sortable 上。如果使用了该选项，一个 draggable 可被放置在一个 sortable 列表上，然后成为列表的一部分。注意： `helper` 选项必须设置为 `"clone"` ，以便更好地工作。必须包含 可排序小部件（Sortable Widget）。

代码实例：

初始化带有指定 `connectToSortable` 选项的 draggable：

```
$( ".selector" ).draggable({ connectToSortable: "#my-sortable" });
```

在初始化后，获取或设置 `connectToSortable` 选项：

```
// getter
var connectToSortable = $( ".selector" ).draggable( "option", "conn

// setter
$( ".selector" ).draggable( "option", "connectToSortable", "#my-so
```

默认值：false

containment

类型：Selector 或 Element 或 String 或 Array

描述：约束在指定元素或区域的边界内拖拽。

支持多个类型：

- **Selector**：可拖拽元素将被包含在 selector 第一个元素的边界内。如果未找到元素，则不设置 containment。
- **Element**：可拖拽元素将被韩寒在元素的边界。
- **String**：可能的值：`"parent"`、`"document"`、`"window"`。
- **Array**：一个数组，以形式 `[ x1, y1, x2, y2 ]` 定义元素的边界。

代码实例：

初始化带有指定 `containment` 选项的 draggable：

```
$( ".selector" ).draggable({ containment: "parent" });
```

在初始化后，获取或设置 `containment` 选项：

```
// getter
var containment = $( ".selector" ).draggable( "option", "containmen

// setter
$( ".selector" ).draggable( "option", "containment", "parent" );
```

默认值：false

cursor

类型：String

描述：拖拽操作期间的 CSS 光标。

代码实例：

初始化带有指定 `cursor` 选项的 draggable：

```
$( ".selector" ).draggable({ cursor: "crosshair" });
```

在初始化后，获取或设置 `cursor` 选项：

```
// getter
var cursor = $( ".selector" ).draggable( "option", "cursor" );

// setter
$( ".selector" ).draggable( "option", "cursor", "crosshair" );
```

默认值："auto"

cursorAt

类型：Object

描述：设置拖拽助手（helper）相对于鼠标光标的偏移。坐标可通过一个或两个键的组合成一个哈希给出：`{ top, left, right, bottom }`。

代码实例：

初始化带有指定 `cursorAt` 选项的 draggable：

```
$( ".selector" ).draggable({ cursorAt: { left: 5 } });
```

在初始化后，获取或设置 `cursorAt` 选项：

```
// getter
var cursorAt = $( ".selector" ).draggable( "option", "cursorAt" );

// setter
$( ".selector" ).draggable( "option", "cursorAt", { left: 5 } );
```

默认值：false

delay

类型：Number

描述：鼠标按下后直到拖拽开始为止的时间，以毫秒计。该选项可以防止点击在某个元素上时不必要的拖拽。

代码实例：

初始化带有指定 `delay` 选项的 draggable：

```
$( ".selector" ).draggable({ delay: 300 });
```

在初始化后，获取或设置 `delay` 选项：

```
// getter
var delay = $( ".selector" ).draggable( "option", "delay" );

// setter
$( ".selector" ).draggable( "option", "delay", 300 );
```

默认值：0

disabled

类型：Boolean

描述：如果设置为 `true`，则禁用该 draggable。

代码实例：

初始化带有指定 `disabled` 选项的 draggable：

```
$( ".selector" ).draggable({ disabled: true });
```

在初始化后，获取或设置 `disabled` 选项：

```
// getter
var disabled = $( ".selector" ).draggable( "option", "disabled" );

// setter
$( ".selector" ).draggable( "option", "disabled", true );
```

默认值：false

distance

类型：Number

描述：鼠标按下后拖拽开始前必须移动的距离，以像素计。该选项可以防止点击在某个元素上时不必要的拖拽。

代码实例：

初始化带有指定 `distance` 选项的 draggable：

```
$( ".selector" ).draggable({ distance: 10 });
```

在初始化后，获取或设置 `distance` 选项：

```
// getter
var distance = $( ".selector" ).draggable( "option", "distance" );

// setter
$( ".selector" ).draggable( "option", "distance", 10 );
```

默认值：1

grid

类型：Array

描述：对齐拖拽助手（helper）到网格，每个 x 和 y 像素。数组形式必须是 `[ x, y ]`。

代码实例：

初始化带有指定 `grid` 选项的 draggable：

```
$( ".selector" ).draggable({ grid: [ 50, 20 ] });
```

在初始化后，获取或设置 `grid` 选项：

```
// getter
var grid = $( ".selector" ).draggable( "option", "grid" );

// setter
$( ".selector" ).draggable( "option", "grid", [ 50, 20 ] );
```

默认值：false

handle

类型：Selector 或 Element

描述：如果指定了该选项，则限制开始拖拽，除非鼠标在指定的元素上按下。只有可拖拽（draggable）元素的后代元素才允许被拖拽。

代码实例：

初始化带有指定 `handle` 选项的 draggable：

```
$( ".selector" ).draggable({ handle: "h2" });
```

在初始化后，获取或设置 `handle` 选项：

```
// getter
var handle = $( ".selector" ).draggable( "option", "handle" );

// setter
$( ".selector" ).draggable( "option", "handle", "h2" );
```

默认值：false

helper

类型：String 或 Function()

描述：允许一个 helper 元素用于拖拽显示。

支持多个类型：

- **String**：如果设置为 `"clone"`，元素将被克隆，且克隆将被拖拽。
- **Function**：一个函数，将返回拖拽时要使用的 DOMElement。

代码实例：

初始化带有指定 `helper` 选项的 draggable：

```
$( ".selector" ).draggable({ helper: "clone" });
```

在初始化后，获取或设置 `helper` 选项：

```
// getter
var helper = $( ".selector" ).draggable( "option", "helper" );

// setter
$( ".selector" ).draggable( "option", "helper", "clone" );
```

默认值："original"

iframeFix

类型：Boolean 或 Selector

描述：防止拖拽期间 iframes 捕捉鼠标移动（mousemove）事件。在与 `cursorAt` 选项结合使用时，或鼠标光标未覆盖在助手（helper）上时，非常有用。

支持多个类型：

- **Boolean**：当设置为 `true` 时，透明遮罩将被放置在页面上所有 iframes 上。
- **Selector**：匹配 selector 的任意 iframes 将被透明遮罩覆盖。

代码实例：

初始化带有指定 `iframeFix` 选项的 draggable：

```
$( ".selector" ).draggable({ iframeFix: true });
```

在初始化后，获取或设置 `iframeFix` 选项：

```
// getter
var iframeFix = $( ".selector" ).draggable( "option", "iframeFix" 

// setter
$( ".selector" ).draggable( "option", "iframeFix", true );
```

默认值：false

opacity

类型：Number

描述：当被拖拽时助手（helper）的不透明度。

代码实例：

初始化带有指定 `opacity` 选项的 draggable：

```
$( ".selector" ).draggable({ opacity: 0.35 });
```

在初始化后，获取或设置 `opacity` 选项：

```
// getter
var opacity = $( ".selector" ).draggable( "option", "opacity" );

// setter
$( ".selector" ).draggable( "option", "opacity", 0.35 );
```

默认值：false

refreshPositions

类型：Boolean

描述：如果设置为 `true`，在每次鼠标移动（mousemove）时都会计算所有可放置的位置。注意：这解决了高度动态的问题，但是明显降低了性能。

代码实例：

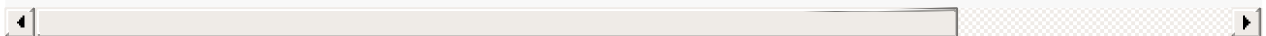
初始化带有指定 `refreshPositions` 选项的 draggable：

```
$( ".selector" ).draggable({ refreshPositions: true });
```

在初始化后，获取或设置 `refreshPositions` 选项：

```
// getter
var refreshPositions = $( ".selector" ).draggable( "option", "refre

// setter
$( ".selector" ).draggable( "option", "refreshPositions", true );
```



默认值：false

revert

类型：Boolean 或 String 或 Function()

描述：当拖拽停止时，元素是否还原到它的开始位置。

支持多个类型：

- **Boolean**：如果设置为 `true`，元素总会还原。
- **String**：如果设置为 `"invalid"`，还原仅在 draggable 未放置在 droppable 上时发生，如果设置为 `"valid"` 则相反。
- **Function**：一个函数，确定元素是否还原到它的开始位置。该函数必须返回 `true` 才能还原元素。

代码实例：

初始化带有指定 `revert` 选项的 draggable：

```
$( ".selector" ).draggable({ revert: true });
```

在初始化后，获取或设置 `revert` 选项：

```
// getter
var revert = $( ".selector" ).draggable( "option", "revert" );

// setter
$( ".selector" ).draggable( "option", "revert", true );
```

默认值：false

revertDuration

类型：Number

描述：还原（revert）动画的持续时间，以毫秒计。如果 `revert` 选项是 `false` 则忽略。

代码实例：

初始化带有指定 `revertDuration` 选项的 draggable：

```
$( ".selector" ).draggable({ revertDuration: 200 });
```

在初始化后，获取或设置 `revertDuration` 选项：

```
// getter
var revertDuration = $( ".selector" ).draggable( "option", "revertD

// setter
$( ".selector" ).draggable( "option", "revertDuration", 200 );
```

默认值：500

scope

类型：String

描述：用于组合配套 draggable 和 droppable 项，除了 droppable 的 `accept` 选项之外。一个与 droppable 带有相同的 `scope` 值的 draggable 会被该 droppable 接受。

代码实例：

初始化带有指定 `scope` 选项的 draggable：

```
$( ".selector" ).draggable({ scope: "tasks" });
```

在初始化后，获取或设置 `scope` 选项：

```
// getter
var scope = $( ".selector" ).draggable( "option", "scope" );

// setter
$( ".selector" ).draggable( "option", "scope", "tasks" );
```

默认值："default"

scroll

类型：Boolean

描述：如果设置为 `true`，当拖拽时容器会自动滚动。

代码实例：

初始化带有指定 `scroll` 选项的 draggable：

```
$( ".selector" ).draggable({ scroll: false });
```

在初始化后，获取或设置 `scroll` 选项：

```
// getter
var scroll = $( ".selector" ).draggable( "option", "scroll" );

// setter
$( ".selector" ).draggable( "option", "scroll", false );
```

默认值：true

scrollSensitivity

类型：Number

描述：从要滚动的视区边缘起的距离，以像素计。距离是相对于指针的，不是相对于 draggable。如果 `scroll` 选项是 `false` 则忽略。

代码实例：

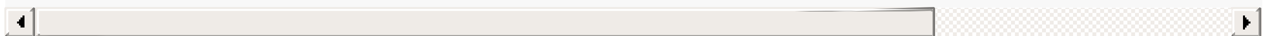
初始化带有指定 `scrollSensitivity` 选项的 draggable：

```
$( ".selector" ).draggable({ scrollSensitivity: 100 });
```

在初始化后，获取或设置 `scrollSensitivity` 选项：

```
// getter
var scrollSensitivity = $( ".selector" ).draggable( "option", "scro

// setter
$( ".selector" ).draggable( "option", "scrollSensitivity", 100 );
```



默认值：20

scrollSpeed

类型：Number

描述：当鼠标指针获取到在 `scrollSensitivity` 距离内时，窗体滚动的速度。如果 `scroll` 选项是 `false` 则忽略。

代码实例：

初始化带有指定 `scrollSpeed` 选项的 draggable：

```
$( ".selector" ).draggable({ scrollSpeed: 100 });
```

在初始化后，获取或设置 `scrollSpeed` 选项：

```
// getter
var scrollSpeed = $( ".selector" ).draggable( "option", "scrollSpee

// setter
$( ".selector" ).draggable( "option", "scrollSpeed", 100 );
```

默认值：20

snap

类型：Boolean 或 Selector

描述：元素是否对齐到其他元素。

支持多个类型：

- **Boolean**：当设置为 `true` 时，元素会对齐到其它可拖拽（draggable ）元素。
- **Selector**：一个选择器，指定要对齐到哪个元素。

代码实例：

初始化带有指定 `snap` 选项的 draggable：

```
$( ".selector" ).draggable({ snap: true });
```

在初始化后，获取或设置 `snap` 选项：

```
// getter
var snap = $( ".selector" ).draggable( "option", "snap" );

// setter
$( ".selector" ).draggable( "option", "snap", true );
```

默认值：false

snapMode

类型：String

描述：决定 draggable 将对齐到对齐元素的哪个边缘。如果 `snap` 选项是 `false` 则忽略。可能的值：`"inner"`、`"outer"`、`"both"`。

代码实例：

初始化带有指定 `snapMode` 选项的 draggable：

```
$( ".selector" ).draggable({ snapMode: "inner" });
```

在初始化后，获取或设置 `snapMode` 选项：

```
// getter
var snapMode = $( ".selector" ).draggable( "option", "snapMode" );

// setter
$( ".selector" ).draggable( "option", "snapMode", "inner" );
```

默认值："both"

snapTolerance

类型：Number

描述：从要发生对齐的对齐元素边缘起的距离，以像素计。如果 `snap` 选项是 `false` 则忽略。

代码实例：

初始化带有指定 `snapTolerance` 选项的 draggable：

```
$( ".selector" ).draggable({ snapTolerance: 30 });
```

在初始化后，获取或设置 `snapTolerance` 选项：

```
// getter
var snapTolerance = $( ".selector" ).draggable( "option", "snapTole

// setter
$( ".selector" ).draggable( "option", "snapTolerance", 30 );
```

默认值：20

stack

类型：Selector

描述：控制匹配选择器（selector）的元素集合的 z-index，总是在当前拖拽项的前面，在类似窗口管理器这样的事物中非常有用。

代码实例：

初始化带有指定 `stack` 选项的 draggable：

```
$( ".selector" ).draggable({ stack: ".products" });
```

在初始化后，获取或设置 `stack` 选项：

```
// getter
var stack = $( ".selector" ).draggable( "option", "stack" );

// setter
$( ".selector" ).draggable( "option", "stack", ".products" );
```

默认值：false

zIndex

类型：Number

描述：当被拖拽时，助手（helper）的 Z-index。

代码实例：

初始化带有指定 `zIndex` 选项的 draggable：

```
$( ".selector" ).draggable({ zIndex: 100 });
```

在初始化后，获取或设置 `zIndex` 选项：

```
// getter
var zIndex = $( ".selector" ).draggable( "option", "zIndex" );

// setter
$( ".selector" ).draggable( "option", "zIndex", 100 );
```

默认值：false

方法

destroy()

类型：jQuery (plugin only)

描述：完全移除 draggable 功能。这会把元素返回到它的预初始化状态。

- 该方法不接受任何参数。

代码实例：

调用 destroy 方法：

```
$( ".selector" ).draggable( "destroy" );
```

disable()

类型：jQuery (plugin only)

描述：禁用 draggable。

- 该方法不接受任何参数。

代码实例：

调用 disable 方法：

```
$( ".selector" ).draggable( "disable" );
```

enable()

类型：jQuery (plugin only)

描述：启用 draggable。

- 该方法不接受任何参数。

代码实例：

调用 enable 方法：

```
$( ".selector" ).draggable( "enable" );
```

option(optionName)

类型：Object

描述：获取当前与指定的 `optionName` 关联的值。

- **optionName** 类型：String 描述：要获取的选项的名称。

代码实例：

调用该方法：

```
var isDisabled = $( ".selector" ).draggable( "option", "disabled" 
```

option()

类型：PlainObject

描述：获取一个包含键/值对的对象，键/值对表示当前 draggable 选项哈希。

- 该方法不接受任何参数。

代码实例：

调用该方法：

```
var options = $( ".selector" ).draggable( "option" );
```

option(optionName, value)

类型：jQuery (plugin only)

描述：设置与指定的 `optionName` 关联的 draggable 选项的值。

- **optionName** 类型：String 描述：要设置的选项的名称。
- **value** 类型：Object 描述：要为选项设置的值。

代码实例：

调用该方法：

```
$( ".selector" ).draggable( "option", "disabled", true );
```

option(options)

类型：jQuery (plugin only)

描述：为 draggable 设置一个或多个选项。

- **options** 类型：Object 描述：要设置的 option-value 对。

代码实例：

调用该方法：

```
$( ".selector" ).draggable( "option", { disabled: true } );
```

widget()

类型：jQuery

描述：返回一个包含 draggable 元素的 `jQuery` 对象。

- 该方法不接受任何参数。

代码实例：

调用 widget 方法：

```
var widget = $( ".selector" ).draggable( "widget" );
```

事件

create(event, ui)

类型：dragcreate

描述：当 draggable 被创建时触发。

- **event** 类型：Event
- **ui** 类型：Object

注意： `ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

代码实例：

初始化带有指定 create 回调的 draggable：

```
$( ".selector" ).draggable({
  create: function( event, ui ) {}
});
```

绑定一个事件监听器到 dragcreate 事件：

```
$( ".selector" ).on( "dragcreate", function( event, ui ) {} );
```

drag(event, ui)

类型：drag

描述：在拖拽期间当鼠标移动时触发。

- **event** 类型：Event
- **ui** 类型：Object
  - **helper** 类型：jQuery 描述：jQuery 对象，表示被拖拽的助手（helper）。
  - **position** 类型：Object 描述：助手（helper）的当前 CSS 位置，比如 `{ top, left }` 对象。
  - **offset** 类型：Object 描述：助手（helper）的当前偏移位置，比如 `{ top, left }` 对象。

代码实例：

初始化带有指定 drag 回调的 draggable：

```
$( ".selector" ).draggable({
  drag: function( event, ui ) {}
});
```

绑定一个事件监听器到 drag 事件：

```
$( ".selector" ).on( "drag", function( event, ui ) {} );
```

start(event, ui)

类型：dragstart

描述：当拖拽开始时触发。

- **event** 类型：Event
- **ui** 类型：Object
  - **helper** 类型：jQuery 描述：jQuery 对象，表示被拖拽的助手（helper）。
  - **position** 类型：Object 描述：助手（helper）的当前 CSS 位置，比如 `{ top, left }` 对象。
  - **offset** 类型：Object 描述：助手（helper）的当前偏移位置，比如 `{ top, left }` 对象。

代码实例：

初始化带有指定 start 回调的 draggable：

```
$( ".selector" ).draggable({
  start: function( event, ui ) {}
});
```

绑定一个事件监听器到 dragstart 事件：

```
$( ".selector" ).on( "dragstart", function( event, ui ) {} );
```

stop(event, ui)

类型：dragstop

描述：当拖拽停止时触发。

- **event** 类型：Event
- **ui** 类型：Object
  - **helper** 类型：jQuery 描述：jQuery 对象，表示被拖拽的助手（helper）。
  - **position** 类型：Object 描述：助手（helper）的当前 CSS 位置，比如 `{ top, left }` 对象。
  - **offset** 类型：Object 描述：助手（helper）的当前偏移位置，比如 `{ top, left }` 对象。

代码实例：

初始化带有指定 stop 回调的 draggable：

```
$( ".selector" ).draggable({
  stop: function( event, ui ) {}
});
```

绑定一个事件监听器到 dragstop 事件：

```
$( ".selector" ).on( "dragstop", function( event, ui ) {} );
```

实例

一个简单的 jQuery UI 可拖拽小部件（Draggable Widget）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>可拖拽小部件（Draggable Widget）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  #draggable {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<div id="draggable">请拖拽我！</div>

<script>
$( "#draggable" ).draggable();
</script>

</body>
</html>
```

jQuery UI API - 可放置小部件（Droppable Widget）

所属类别

交互（Interactions）

用法

描述：为可拖拽小部件创建目标。

版本新增：1.0

依赖：

- UI 核心（UI Core）
- 部件库（Widget Factory）
- 鼠标交互（Mouse Interaction）

注释：jQuery UI 可放置（Droppable）插件让被选元素可放置（意味着它们在拖拽后接受被放置）。您可以逐个指定哪一个 draggable 会接受。

快速导航

选项	
acceptactiveClassaddClassesdisabledgreedyhoverClassscopetolerance	destr

accept

类型：Selector 或 Function()

描述：控制哪个可拖拽（draggable）元素可被 droppable 接受。

支持多个类型：

- **Selector**：一个选择器，指定哪个可拖拽（draggable）元素可被 droppable 接受。
- **Function()**：一个函数，将被页面上每个 draggable 调用（作为第一个参数传递给函数）。如果 draggable 被接受，该函数必须返回 `true` 。

代码实例：

初始化带有指定 `accept` 选项的 droppable：

```
$( ".selector" ).droppable({ accept: ".special" });
```

在初始化后，获取或设置 `accept` 选项：

```
// getter
var accept = $( ".selector" ).droppable( "option", "accept" );

// setter
$( ".selector" ).droppable( "option", "accept", ".special" );
```

默认值："*"

activeClass

类型：String

描述：如果指定了该选项，当一个可接受的 draggable 被拖拽时，class 将被添加到 droppable。

代码实例：

初始化带有指定 `activeClass` 选项的 droppable：

```
$( ".selector" ).droppable({ activeClass: "ui-state-highlight" });
```

在初始化后，获取或设置 `activeClass` 选项：

```
// getter
var activeClass = $( ".selector" ).droppable( "option", "activeClas

// setter
$( ".selector" ).droppable( "option", "activeClass", "ui-state-high
```

默认值：false

addClasses

类型：Boolean

描述：如果设置为 `false`，将防止 `ui-droppable` class 被添加。这在数百个元素上调用 `.droppable()` 时有助于性能优化。

代码实例：

初始化带有指定 `addClasses` 选项的 droppable：

```
$( ".selector" ).droppable({ addClasses: false });
```

在初始化后，获取或设置 `addClasses` 选项：

```
// getter
var addClasses = $( ".selector" ).droppable( "option", "addClasses'

// setter
$( ".selector" ).droppable( "option", "addClasses", false );
```

默认值：true

disabled

类型：Boolean

描述：如果设置为 `true`，则禁用该 droppable。

代码实例：

初始化带有指定 `disabled` 选项的 droppable：

```
$( ".selector" ).droppable({ disabled: true });
```

在初始化后，获取或设置 `disabled` 选项：

```
// getter
var disabled = $( ".selector" ).droppable( "option", "disabled" );

// setter
$( ".selector" ).droppable( "option", "disabled", true );
```

默认值：false

greedy

类型：Boolean

描述：默认情况下，当一个元素被放置在嵌套是 droppable 上时，每个 droppable 将接收该元素。然而，通过设置该选项为 `true`，任何父元素的 droppable 将无法接收该元素。 `drop` 事件仍将照常，但会检查 `event.target` 以便查看哪个

droppable 接收 draggable 元素。

代码实例：

初始化带有指定 `greedy` 选项的 droppable：

```
$( ".selector" ).droppable({ greedy: true });
```

在初始化后，获取或设置 `greedy` 选项：

```
// getter
var greedy = $( ".selector" ).droppable( "option", "greedy" );

// setter
$( ".selector" ).droppable( "option", "greedy", true );
```

默认值：false

hoverClass

类型：String

描述：如果指定了该选项，当一个可接受 draggable 被覆盖在 droppable 上时，class 将被添加到 droppable。

代码实例：

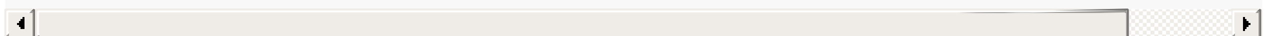
初始化带有指定 `hoverClass` 选项的 droppable：

```
$( ".selector" ).droppable({ hoverClass: "drop-hover" });
```

在初始化后，获取或设置 `hoverClass` 选项：

```
// getter
var hoverClass = $( ".selector" ).droppable( "option", "hoverClass'

// setter
$( ".selector" ).droppable( "option", "hoverClass", "drop-hover" );
```



默认值：false

scope

类型：String

描述：用于组合配套 draggable 和 droppable 项，除了 droppable 的 `accept` 选项之外。一个与 droppable 带有相同的 `scope` 值的 draggable 会被该 droppable 接受。

代码实例：

初始化带有指定 `scope` 选项的 droppable：

```
$( ".selector" ).droppable({ scope: "tasks" });
```

在初始化后，获取或设置 `scope` 选项：

```
// getter
var scope = $( ".selector" ).droppable( "option", "scope" );

// setter
$( ".selector" ).droppable( "option", "scope", "tasks" );
```

默认值："default"

tolerance

类型：String

描述：指定用于测试一个 draggable 是否覆盖在一个 droppable 上的模式。可能的值：

- `"fit"` ：draggable 完全重叠在 droppable 上。
- `"intersect"` ：draggable 重叠在 droppable 上，两个方向上至少 50%。
- `"pointer"` ：鼠标指针重叠在 droppable 上。
- `"touch"` ：draggable 重叠在 droppable 上，任何数量皆可。

代码实例：

初始化带有指定 `tolerance` 选项的 droppable：

```
$( ".selector" ).droppable({ tolerance: "fit" });
```

在初始化后，获取或设置 `tolerance` 选项：

```
// getter
var tolerance = $( ".selector" ).droppable( "option", "tolerance" 

// setter
$( ".selector" ).droppable( "option", "tolerance", "fit" );
```

默认值："intersect"

方法

destroy()

类型：jQuery (plugin only)

描述：完全移除 droppable 功能。这会把元素返回到它的预初始化状态。

- 该方法不接受任何参数。

代码实例：

调用 destroy 方法：

```
$( ".selector" ).droppable( "destroy" );
```

disable()

类型：jQuery (plugin only)

描述：禁用 droppable。

- 该方法不接受任何参数。

代码实例：

调用 disable 方法：

```
$( ".selector" ).droppable( "disable" );
```

enable()

类型：jQuery (plugin only)

描述：启用 droppable。

- 该方法不接受任何参数。

代码实例：

调用 enable 方法：

```
$( ".selector" ).droppable( "enable" );
```

option(optionName)

类型：Object

描述：获取当前与指定的 `optionName` 关联的值。

- **optionName** 类型：String 描述：要获取的选项的名称。

代码实例：

调用该方法：

```
var isDisabled = $( ".selector" ).droppable( "option", "disabled" )
```

option()

类型：PlainObject

描述：获取一个包含键/值对的对象，键/值对表示当前 droppable 选项哈希。

- 该方法不接受任何参数。

代码实例：

调用该方法：

```
var options = $( ".selector" ).droppable( "option" );
```

option(optionName, value)

类型：jQuery (plugin only)

描述：设置与指定的 `optionName` 关联的 droppable 选项的值。

- **optionName** 类型：String 描述：要设置的选项的名称。
- **value** 类型：Object 描述：要为选项设置的值。

代码实例：

调用该方法：

```
$( ".selector" ).droppable( "option", "disabled", true );
```

option(options)

类型：jQuery (plugin only)

描述：为 droppable 设置一个或多个选项。

- **options** 类型：Object 描述：要设置的 option-value 对。

代码实例：

调用该方法：

```
$( ".selector" ).droppable( "option", { disabled: true } );
```

widget()

类型：jQuery

描述：返回一个包含 droppable 元素的 `jQuery` 对象。

- 该方法不接受任何参数。

代码实例：

调用 widget 方法：

```
var widget = $( ".selector" ).droppable( "widget" );
```

事件

activate(event, ui)

类型：dropactivate

描述：当一个可接受的 draggable 开始拖拽时触发。如果您想让 droppable 被放置时"点亮"，该选项就可以派上用场。

- **event** 类型：Event
- **ui** 类型：Object
  - **draggable** 类型：jQuery 描述：jQuery 对象，表示 draggable 元素。
  - **helper** 类型：jQuery 描述：jQuery 对象，表示被拖拽的助手（helper）。
  - **position** 类型：Object 描述：draggable 助手（helper）的当前 CSS 位置，比如 `{ top, left }` 对象。
  - **offset** 类型：Object 描述：draggable 助手（helper）的当前偏移位置，比如 `{ top, left }` 对象。

代码实例：

初始化带有指定 activate 回调的 droppable：

```
$( ".selector" ).droppable({
  activate: function( event, ui ) {}
});
```

绑定一个事件监听器到 dropactivate 事件：

```
$( ".selector" ).on( "dropactivate", function( event, ui ) {} );
```

create(event, ui)

类型：dropcreate

描述：当 droppable 被创建时触发。

- **event** 类型：Event
- **ui** 类型：Object

注意：`ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

代码实例：

初始化带有指定 create 回调的 droppable：

```
$( ".selector" ).droppable({
  create: function( event, ui ) {}
});
```

绑定一个事件监听器到 dropcreate 事件：

```
$( ".selector" ).on( "dropcreate", function( event, ui ) {} );
```

deactivate(event, ui)

类型：dropdeactivate

描述：当一个可接受的 draggable 停止拖拽时触发。

- **event** 类型：Event
- **ui** 类型：Object
  - **draggable** 类型：jQuery 描述：jQuery 对象，表示 draggable 元素。
  - **helper** 类型：jQuery 描述：jQuery 对象，表示被拖拽的助手（helper）。
  - **position** 类型：Object 描述：draggable 助手（helper）的当前 CSS 位置，比如 `{ top, left }` 对象。
  - **offset** 类型：Object 描述：draggable 助手（helper）的当前偏移位置，

比如 `{ top, left }` 对象。

代码实例：

初始化带有指定 deactivate 回调的 droppable：

```
$( ".selector" ).droppable({
  deactivate: function( event, ui ) {}
});
```

绑定一个事件监听器到 dropdeactivate 事件：

```
$( ".selector" ).on( "dropdeactivate", function( event, ui ) {} );
```

drop(event, ui)

类型：drop

描述：当一个可接受的 draggable 被放置在 droppable（基于 `[tolerance](#option-tolerance)` 选项）上时触发。

- **event** 类型：Event
- **ui** 类型：Object
  - **draggable** 类型：jQuery 描述：jQuery 对象，表示 draggable 元素。
  - **helper** 类型：jQuery 描述：jQuery 对象，表示被拖拽的助手（helper）。
  - **position** 类型：Object 描述：draggable 助手（helper）的当前 CSS 位置，比如 `{ top, left }` 对象。
  - **offset** 类型：Object 描述：draggable 助手（helper）的当前偏移位置，比如 `{ top, left }` 对象。

代码实例：

初始化带有指定 drop 回调的 droppable：

```
$( ".selector" ).droppable({
  drop: function( event, ui ) {}
});
```

绑定一个事件监听器到 drop 事件：

```
$( ".selector" ).on( "drop", function( event, ui ) {} );
```

out(event, ui)

类型：dropout

描述：当 droppable 被拖拽出 droppable（基于 `[tolerance](#option-tolerance)` 选项）时触发。

- **event** 类型：Event
- **ui** 类型：Object

注意：`ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

代码实例：

初始化带有指定 out 回调的 droppable：

```
$( ".selector" ).droppable({
  out: function( event, ui ) {}
});
```

绑定一个事件监听器到 dropout 事件：

```
$( ".selector" ).on( "dropout", function( event, ui ) {} );
```

over(event, ui)

类型：dropover

描述：当一个可接受的 draggable 被拖拽在 droppable（基于 `[tolerance](#option-tolerance)` 选项）上时触发。

- **event** 类型：Event
- **ui** 类型：Object
  - **draggable** 类型：jQuery 描述：jQuery 对象，表示 draggable 元素。
  - **helper** 类型：jQuery 描述：jQuery 对象，表示被拖拽的助手（helper）。
  - **position** 类型：Object 描述：draggable 助手（helper）的当前 CSS 位置，比如 `{ top, left }` 对象。
  - **offset** 类型：Object 描述：draggable 助手（helper）的当前偏移位置，比如 `{ top, left }` 对象。

代码实例：

初始化带有指定 over 回调的 droppable：

```
$( ".selector" ).droppable({
  over: function( event, ui ) {}
});
```

绑定一个事件监听器到 dropover 事件：

```
$( ".selector" ).on( "dropover", function( event, ui ) {} );
```

实例

一对 draggable 和 droppable 元素。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>可放置小部件（Droppable Widget）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  #draggable {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  #droppable {
    position: absolute;
    left: 250px;
    top: 0;
    width: 125px;
    height: 125px;
    background: #999;
    color: #fff;
    padding: 10px;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<div id="droppable">请放置到这里！</div>
<div id="draggable">请拖拽我！</div>

<script>
$( "#draggable" ).draggable();
$( "#droppable" ).droppable({
  drop: function() {
    alert( "dropped" );
  }
});
</script>

</body>
</html>
```

jQuery UI API - 鼠标交互（Mouse Interaction）

所属类别

交互（Interactions） | 实用工具（Utilities）

用法

描述：基本交互层。

依赖：

- 部件库（Widget Factory）

注释：与 `jQuery.Widget` 相似，鼠标交互的目的不是直接使用。这是一个纯粹给其他小部件继承用的基础层。该页面有添加到 `jQuery.Widget` 的文档，但是它包含了不能被覆盖的内部方法。公共的 API 是 `_mouseStart()`、`_mouseDrag()`、`_mouseStop()` 和 `_mouseCapture()`。

快速导航

选项	
canceldelaydistance	_mouseCapture_mouseDelayMet_mouseDestroy_mouse

cancel

类型：Selector

描述：防止从指定的元素上开始交互。

代码实例：

初始化带有指定 `cancel` 选项的 mouse：

```
$( ".selector" ).mouse({ cancel: ".title" });
```

在初始化后，获取或设置 `cancel` 选项：

```
// getter
var cancel = $( ".selector" ).mouse( "option", "cancel" );

// setter
$( ".selector" ).mouse( "option", "cancel", ".title" );
```

默认值："input, textarea, button, select, option"

delay

类型：Number

描述：鼠标按下后直至交互开始的事件，以毫秒计。该选项可用于防止点击在一个元素上时不必要的交互。

代码实例：

初始化带有指定 `delay` 选项的 mouse：

```
$( ".selector" ).mouse({ delay: 300 });
```

在初始化后，获取或设置 `delay` 选项：

```
// getter
var delay = $( ".selector" ).mouse( "option", "delay" );

// setter
$( ".selector" ).mouse( "option", "delay", 300 );
```

默认值：0

distance

类型：Number

描述：鼠标按下后交互开始前鼠标必须移动的距离，以像素计。该选项可用于防止点击在一个元素上时不必要的交互。

代码实例：

初始化带有指定 `distance` 选项的 mouse：

```
$( ".selector" ).mouse({ distance: 10 });
```

在初始化后，获取或设置 `distance` 选项：

```
// getter
var distance = $( ".selector" ).mouse( "option", "distance" );

// setter
$( ".selector" ).mouse( "option", "distance", 10 );
```

默认值：1

方法

_mouseCapture()

类型：Boolean

描述：决定交互是否应该基于交互的事件目标开始。默认的实现总是返回 `true` 。

- 该方法不接受任何参数。

代码实例：

调用 _mouseCapture 方法：

```
$( ".selector" ).mouse( "_mouseCapture" );
```

_mouseDelayMet()

类型：Boolean

描述：决定 `[delay](#option-delay)` 选项是否满足当前交互。

- 该方法不接受任何参数。

代码实例：

调用 _mouseDelayMet 方法：

```
$( ".selector" ).mouse( "_mouseDelayMet" );
```

_mouseDestroy()

类型：jQuery (plugin only)

描述：销毁交互事件处理程序。这必须调用来自扩展的小部件的 `_destroy()` 方法。

- 该方法不接受任何参数。

代码实例：

调用 _mouseDestroy 方法：

```
$( ".selector" ).mouse( "_mouseDestroy" );
```

_mouseDistanceMet()

类型：Boolean

描述：决定 `[distance](#option-distance)` 选项是否满足当前交互。

- 该方法不接受任何参数。

代码实例：

调用 _mouseDistanceMet 方法：

```
$( ".selector" ).mouse( "_mouseDistanceMet" );
```

_mouseDown()

类型：jQuery (plugin only)

描述：处理交互的开始。确认与主要的鼠标按钮关联的事件，确保 `[delay](#option-delay)` 与 `[distance](#option-distance)` 在交互启动之前得到满足。当交互已经准备开始，为要处理的扩展小部件调用 `[_mouseStart](#method-_mouseStart)` 方法。

- 该方法不接受任何参数。

代码实例：

调用 _mouseDown 方法：

```
$( ".selector" ).mouse( "_mouseDown" );
```

_mouseDrag()

类型：jQuery (plugin only)

描述：扩展小部件应实现一个 `_mouseDrag()` 方法，来处理交互的每个移动。该方法将接收与鼠标移动相关联的鼠标事件。

- 该方法不接受任何参数。

代码实例：

调用 _mouseDrag 方法：

```
$( ".selector" ).mouse( "_mouseDrag" );
```

_mouseInit()

类型：jQuery (plugin only)

描述：初始化交互事件处理程序。这必须调用来自扩展的小部件的 `_create()` 方法。

- 该方法不接受任何参数。

代码实例：

调用 _mouseInit 方法：

```
$( ".selector" ).mouse( "_mouseInit" );
```

_mouseMove()

类型：jQuery (plugin only)

描述：处理交互的每个移动。为要处理的扩展小部件调用 _mouseDrag 方法。

- 该方法不接受任何参数。

代码实例：

调用 _mouseMove 方法：

```
$( ".selector" ).mouse( "_mouseMove" );
```

_mouseStart()

类型：jQuery (plugin only)

描述：扩展小部件应实现一个 `_mouseStart()` 方法，来处理交互的开始。该方法将接收与交互开始相关联的鼠标事件。

- 该方法不接受任何参数。

代码实例：

调用 _mouseStart 方法：

```
$( ".selector" ).mouse( "_mouseStart" );
```

_mouseStop()

类型：jQuery (plugin only)

描述：扩展小部件应实现一个 `_mouseStop()` 方法，来处理交互的结束。该方法将接收与交互结束相关联的鼠标事件。

- 该方法不接受任何参数。

代码实例：

调用 _mouseStop 方法：

```
$( ".selector" ).mouse( "_mouseStop" );
```

_mouseUp()

类型：jQuery (plugin only)

描述：处理交互的结束。为要处理的扩展小部件调用 _mouseStop 方法。

- 该方法不接受任何参数。

代码实例：

调用 _mouseUp 方法：

```
$( ".selector" ).mouse( "_mouseUp" );
```

jQuery UI API - 可调整尺寸小部件（Resizable Widget）

所属类别

交互（Interactions）

用法

描述：使用鼠标改变元素的尺寸。

版本新增：1.0

依赖：

- UI 核心（UI Core）
- 部件库（Widget Factory）
- 鼠标交互（Mouse Interaction）

注释：jQuery UI 可调整尺寸（Resizable）插件让被选元素可调整尺寸（意味着它们有可拖拽的调整大小的手柄）。您可以指定一个或多个手柄，也可以指定宽度和高度的最小值也最大值。

附加说明：该部件要求一些功能性的 CSS，否则将无法工作。如果您创建了一个自定义的主题，请使用小部件指定的 CSS 文件作为起点。

快速导航

alsoResizeanimateanimateDurationanimateEasingaspectRatioautoHidecancelco

alsoResize

类型：Selector 或 jQuery 或 Element

描述：一个或多个通过 resizable 元素进行同步调整尺寸的元素。

代码实例：

初始化带有指定 `alsoResize` 选项的 resizable：

```
$( ".selector" ).resizable({ alsoResize: "#mirror" });
```

在初始化后，获取或设置 `alsoResize` 选项：

```
// getter
var alsoResize = $( ".selector" ).resizable( "option", "alsoResize'

// setter
$( ".selector" ).resizable( "option", "alsoResize", "#mirror" );
```

默认值：false

animate

类型：Boolean

描述：调整尺寸后动态变化到最终尺寸。

代码实例：

初始化带有指定 `animate` 选项的 resizable：

```
$( ".selector" ).resizable({ animate: true });
```

在初始化后，获取或设置 `animate` 选项：

```
// getter
var animate = $( ".selector" ).resizable( "option", "animate" );

// setter
$( ".selector" ).resizable( "option", "animate", true );
```

默认值：false

animateDuration

类型：Number 或 String

描述：当使用 `[animate](#option-animate)` 选项时，动画持续的时间。

支持多个类型：

- **Number**：持续时间，以毫秒计。
- **String**：一个命名的持续时间，比如 `"slow"` 或 `"fast"` 。

代码实例：

初始化带有指定 `animateDuration` 选项的 resizable：

```
$( ".selector" ).resizable({ animateDuration: "fast" });
```

在初始化后，获取或设置 `animateDuration` 选项：

```
// getter
var animateDuration = $( ".selector" ).resizable( "option", "animat

// setter
$( ".selector" ).resizable( "option", "animateDuration", "fast" );
```

默认值："slow"

animateEasing

类型：String

描述：当使用 `[animate](#option-animate)` 选项时要使用的 Easings。

代码实例：

初始化带有指定 `animateEasing` 选项的 resizable：

```
$( ".selector" ).resizable({ animateEasing: "easeOutBounce" });
```

在初始化后，获取或设置 `animateEasing` 选项：

```
// getter
var animateEasing = $( ".selector" ).resizable( "option", "animateE

// setter
$( ".selector" ).resizable( "option", "animateEasing", "easeOutBour
```

默认值："swing"

aspectRatio

类型：Boolean 或 Number

描述：元素是否应该被限制在一个特定的长宽比。

支持多个类型：

- **Boolean**：当设置为 `true` 时，元素将保持其原有的长宽比。

- **Number**：强制在调整尺寸时元素保持特定的长宽比。

代码实例：

初始化带有指定 `aspectRatio` 选项的 resizable：

```
$( ".selector" ).resizable({ aspectRatio: true });
```

在初始化后，获取或设置 `aspectRatio` 选项：

```
// getter
var aspectRatio = $( ".selector" ).resizable( "option", "aspectRat

// setter
$( ".selector" ).resizable( "option", "aspectRatio", true );
```

默认值：false

autoHide

类型：Boolean

描述：当用户鼠标没有悬浮在元素上时是否隐藏手柄。

代码实例：

初始化带有指定 `autoHide` 选项的 resizable：

```
$( ".selector" ).resizable({ autoHide: true });
```

在初始化后，获取或设置 `autoHide` 选项：

```
// getter
var autoHide = $( ".selector" ).resizable( "option", "autoHide" );

// setter
$( ".selector" ).resizable( "option", "autoHide", true );
```

默认值：false

cancel

类型：Selector

描述：防止从指定的元素上开始调整尺寸。

代码实例：

初始化带有指定 `cancel` 选项的 resizable：

```
$( ".selector" ).resizable({ cancel: ".cancel" });
```

在初始化后，获取或设置 `cancel` 选项：

```
// getter
var cancel = $( ".selector" ).resizable( "option", "cancel" );

// setter
$( ".selector" ).resizable( "option", "cancel", ".cancel" );
```

默认值："input, textarea, button, select, option"

containment

类型：Selector 或 Element 或 String

描述：约束在指定元素或区域的边界内调整尺寸。

支持多个类型：

- **Selector**：可调整尺寸元素将被包含在 selector 第一个元素的边界内。如果未找到元素，则不设置 containment。
- **Element**：可调整尺寸元素将被包含在元素的边界内。
- **String**：可能的值： `"parent"` 、 `"document"` 。

代码实例：

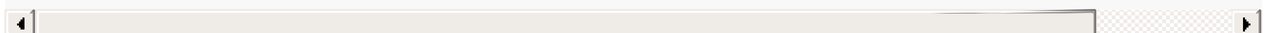
初始化带有指定 `containment` 选项的 resizable：

```
$( ".selector" ).resizable({ containment: "parent" });
```

在初始化后，获取或设置 `containment` 选项：

```
// getter
var containment = $( ".selector" ).resizable( "option", "containmen

// setter
$( ".selector" ).resizable( "option", "containment", "parent" );
```



默认值：false

delay

类型：Number

描述：鼠标按下后直到调整尺寸开始为止的时间，以毫秒计。如果指定了该选项，调整只有在鼠标移动超过时间后才开始。该选项可以防止点击在某个元素上时不必要的调整尺寸。

代码实例：

初始化带有指定 `delay` 选项的 resizable：

```
$( ".selector" ).resizable({ delay: 150 });
```

在初始化后，获取或设置 `delay` 选项：

```
// getter
var delay = $( ".selector" ).resizable( "option", "delay" );

// setter
$( ".selector" ).resizable( "option", "delay", 150 );
```

默认值：0

disabled

类型：Boolean

描述：如果设置为 `true` ，则禁用该 resizable。

代码实例：

初始化带有指定 `disabled` 选项的 resizable：

```
$( ".selector" ).resizable({ disabled: true });
```

在初始化后，获取或设置 `disabled` 选项：

```
// getter
var disabled = $( ".selector" ).resizable( "option", "disabled" );

// setter
$( ".selector" ).resizable( "option", "disabled", true );
```

默认值：false

distance

类型：Number

描述：鼠标按下后调整尺寸开始前必须移动的距离，以像素计。如果指定了该选项，调整只有在鼠标移动超过距离后才开始。该选项可以防止点击在某个元素上时不必要的调整尺寸。

代码实例：

初始化带有指定 `distance` 选项的 resizable：

```
$( ".selector" ).resizable({ distance: 30 });
```

在初始化后，获取或设置 `distance` 选项：

```
// getter
var distance = $( ".selector" ).resizable( "option", "distance" );

// setter
$( ".selector" ).resizable( "option", "distance", 30 );
```

默认值：1

ghost

类型：Boolean

描述：如果设置为 `true`，则为调整尺寸显示一个半透明的辅助元素。

代码实例：

初始化带有指定 `ghost` 选项的 resizable：

```
$( ".selector" ).resizable({ ghost: true });
```

在初始化后，获取或设置 `ghost` 选项：

```
// getter
var ghost = $( ".selector" ).resizable( "option", "ghost" );

// setter
$( ".selector" ).resizable( "option", "ghost", true );
```

默认值：false

grid

类型：Array

描述：把可调整尺寸元素对齐到网格，每个 x 和 y 像素。数组形式必须是 `[ x, y ]`。

代码实例：

初始化带有指定 `grid` 选项的 resizable：

```
$( ".selector" ).resizable({ grid: [ 20, 10 ] });
```

在初始化后，获取或设置 `grid` 选项：

```
// getter
var grid = $( ".selector" ).resizable( "option", "grid" );

// setter
$( ".selector" ).resizable( "option", "grid", [ 20, 10 ] );
```

默认值：false

handles

类型：String 或 Object

描述：可用于调整尺寸的处理程序。

支持多个类型：

- **String**：一个逗号分隔的列表，列表值为下面所列出的任意值：n, e, s, w, ne, se, sw, nw, all。必要的处理程序由插件自动生成。
- **Object**：支持下面的键：{ n, e, s, w, ne, se, sw, nw }。任何指定的值应该是一个匹配作为处理程序使用的 resizable 的子元素的 jQuery 选择器。

注释：当生成您自己的处理程序时，每个处理程序必须有 `ui-resizable-handle` class，也可以是适当的 `appropriate ui-resizable-{direction}` class，比如 ui-resizable-s。

代码实例：

初始化带有指定 `handles` 选项的 resizable：

```
$( ".selector" ).resizable({ handles: "n, e, s, w" });
```

在初始化后，获取或设置 `handles` 选项：

```
// getter
var handles = $( ".selector" ).resizable( "option", "handles" );

// setter
$( ".selector" ).resizable( "option", "handles", "n, e, s, w" );
```

默认值："e, s, se"

helper

类型：String

描述：一个将被添加到代理元素的 class 名称，用于描绘调整手柄拖拽过程中调整的轮廓。一旦调整完成，原来的元素则被重新定义尺寸。

代码实例：

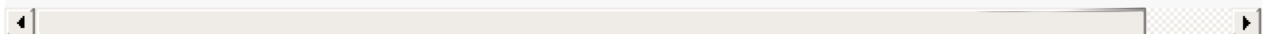
初始化带有指定 `helper` 选项的 resizable：

```
$( ".selector" ).resizable({ helper: "resizable-helper" });
```

在初始化后，获取或设置 `helper` 选项：

```
// getter
var helper = $( ".selector" ).resizable( "option", "helper" );

// setter
$( ".selector" ).resizable( "option", "helper", "resizable-helper"
```



默认值：false

maxHeight

类型：Number

描述：resizable 允许被调整到的最大高度。

代码实例：

初始化带有指定 `maxHeight` 选项的 resizable：

```
$( ".selector" ).resizable({ maxHeight: 300 });
```

在初始化后，获取或设置 `maxHeight` 选项：

```
// getter
var maxHeight = $( ".selector" ).resizable( "option", "maxHeight" )

// setter
$( ".selector" ).resizable( "option", "maxHeight", 300 );
```

默认值：null

maxWidth

类型：Number

描述：resizable 允许被调整到的最大宽度。

代码实例：

初始化带有指定 `maxWidth` 选项的 resizable：

```
$( ".selector" ).resizable({ maxWidth: 300 });
```

在初始化后，获取或设置 `maxWidth` 选项：

```
// getter
var maxWidth = $( ".selector" ).resizable( "option", "maxWidth" );

// setter
$( ".selector" ).resizable( "option", "maxWidth", 300 );
```

默认值：null

minHeight

类型：Number

描述：resizable 允许被调整到的最小高度。

代码实例：

初始化带有指定 `minHeight` 选项的 resizable：

```
$( ".selector" ).resizable({ minHeight: 150 });
```

在初始化后，获取或设置 `minHeight` 选项：

```
// getter
var minHeight = $( ".selector" ).resizable( "option", "minHeight" );

// setter
$( ".selector" ).resizable( "option", "minHeight", 150 );
```

默认值：10

minWidth

类型：Number

描述：resizable 允许被调整到的最小宽度。

代码实例：

初始化带有指定 `minWidth` 选项的 resizable：

```
$( ".selector" ).resizable({ minWidth: 150 });
```

在初始化后，获取或设置 `minWidth` 选项：

```
// getter
var minWidth = $( ".selector" ).resizable( "option", "minWidth" );

// setter
$( ".selector" ).resizable( "option", "minWidth", 150 );
```

默认值：10

方法

destroy()

类型：jQuery (plugin only)

描述：完全移除 resizable 功能。这会把元素返回到它的预初始化状态。

- 该方法不接受任何参数。

代码实例：

调用 destroy 方法：

```
$( ".selector" ).resizable( "destroy" );
```

disable()

类型：jQuery (plugin only)

描述：禁用 resizable。

- 该方法不接受任何参数。

代码实例：

调用 disable 方法：

```
$( ".selector" ).resizable( "disable" );
```

enable()

类型：jQuery (plugin only)

描述：启用 resizable。

- 该方法不接受任何参数。

代码实例：

调用 enable 方法：

```
$( ".selector" ).resizable( "enable" );
```

option(optionName)

类型：Object

描述：获取当前与指定的 `optionName` 关联的值。

- **optionName** 类型：String 描述：要获取的选项的名称。

代码实例：

调用该方法：

```
var isDisabled = $( ".selector" ).resizable( "option", "disabled" )
```

option()

类型：PlainObject

描述：获取一个包含键/值对的对象，键/值对表示当前 resizable 选项哈希。

- 该方法不接受任何参数。

代码实例：

调用该方法：

```
var options = $( ".selector" ).resizable( "option" );
```

option(optionName, value)

类型：jQuery (plugin only)

描述：设置与指定的 `optionName` 关联的 resizable 选项的值。

- **optionName** 类型：String 描述：要设置的选项的名称。
- **value** 类型：Object 描述：要为选项设置的值。

代码实例：

调用该方法：

```
$( ".selector" ).resizable( "option", "disabled", true );
```

option(options)

类型：jQuery (plugin only)

描述：为 resizable 设置一个或多个选项。

- **options** 类型：Object 描述：要设置的 option-value 对。

代码实例：

调用该方法：

```
$( ".selector" ).resizable( "option", { disabled: true } );
```

widget()

类型：jQuery

描述：返回一个包含 resizable 元素的 `jQuery` 对象。

- 该方法不接受任何参数。

代码实例：

调用 widget 方法：

```
var widget = $( ".selector" ).resizable( "widget" );
```

事件

create(event, ui)

类型：resizecreate

描述：当 resizable 被创建时触发。

- **event** 类型：Event
- **ui** 类型：Object

注意： `ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

代码实例：

初始化带有指定 create 回调的 resizable：

```
$( ".selector" ).resizable({
  create: function( event, ui ) {}
});
```

绑定一个事件监听器到 resizecreate 事件：

```
$( ".selector" ).on( "resizecreate", function( event, ui ) {} );
```

resize(event, ui)

类型：resize

描述：在调整尺寸期间当调整手柄拖拽时触发。

- **event** 类型：Event
- **ui** 类型：Object
    - **element** 类型：jQuery 描述：jQuery 对象，表示要被调整尺寸的元素。
    - **helper** 类型：jQuery 描述：jQuery 对象，表示被调整尺寸的助手（helper）。
    - **originalElement** 类型：jQuery 描述：jQuery 对象，表示被包裹之前的原始元素。
    - **originalPosition** 类型：Object 描述：resizable 调整前的位置，表示为 `{ top, left }`。
    - **originalSize** 类型：Object 描述：resizable 调整前的尺寸，表示为 `{ width, height }`。
    - **position** 类型：Object 描述：当前位置，表示为 `{ top, left }`。
    - **size** 类型：Object 描述：当前位置，表示为 `{ width, height }`。

代码实例：

初始化带有指定 resize 回调的 resizable：

```
$( ".selector" ).resizable({
  resize: function( event, ui ) {}
});
```

绑定一个事件监听器到 resize 事件：

```
$( ".selector" ).on( "resize", function( event, ui ) {} );
```

start(event, ui)

类型：resizestart

描述：当调整尺寸操作开始时触发。

- **event** 类型：Event
- **ui** 类型：Object
    - **element** 类型：jQuery 描述：jQuery 对象，表示要被调整尺寸的元素。
    - **helper** 类型：jQuery 描述：jQuery 对象，表示被调整尺寸的助手（helper）。
    - **originalElement** 类型：jQuery 描述：jQuery 对象，表示被包裹之前的原始元素。
    - **originalPosition** 类型：Object 描述：resizable 调整前的位置，表示为 `{ top, left }`。
    - **originalSize** 类型：Object 描述：resizable 调整前的尺寸，表示为 `{ width, height }`。

- ○ **position** 类型：Object 描述：当前位置，表示为 `{ top, left }` 。
- ○ **size** 类型：Object 描述：当前位置，表示为 `{ width, height }` 。

代码实例：

初始化带有指定 start 回调的 resizable：

```
$( ".selector" ).resizable({
  start: function( event, ui ) {}
});
```

绑定一个事件监听器到 resizestart 事件：

```
$( ".selector" ).on( "resizestart", function( event, ui ) {} );
```

stop(event, ui)

类型：resizestop

描述：当调整尺寸操作停止时触发。

- **event** 类型：Event
- **ui** 类型：Object
    - **element** 类型：jQuery 描述：jQuery 对象，表示要被调整尺寸的元素。
    - **helper** 类型：jQuery 描述：jQuery 对象，表示被调整尺寸的助手（helper）。
    - **originalElement** 类型：jQuery 描述：jQuery 对象，表示被包裹之前的原始元素。
    - **originalPosition** 类型：Object 描述：resizable 调整前的位置，表示为 `{ top, left }` 。
    - **originalSize** 类型：Object 描述：resizable 调整前的尺寸，表示为 `{ width, height }` 。
    - **position** 类型：Object 描述：当前位置，表示为 `{ top, left }` 。
    - **size** 类型：Object 描述：当前位置，表示为 `{ width, height }` 。

代码实例：

初始化带有指定 stop 回调的 resizable：

```
$( ".selector" ).resizable({
  stop: function( event, ui ) {}
});
```

绑定一个事件监听器到 resizestop 事件：

```
$( ".selector" ).on( "resizestop", function( event, ui ) {} );
```

实例

一个简单的 jQuery UI 可调整尺寸小部件（Resizable Widget）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>可调整尺寸小部件（Resizable Widget）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  #resizable {
    width: 100px;
    height: 100px;
    background: #ccc;
}    </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<div id="resizable"></div>

<script>
$( "#resizable" ).resizable();
</script>

</body>
</html>
```

jQuery UI API - 可选择小部件（Selectable Widget）

所属类别

交互（Interactions）

用法

描述：使用鼠标选择单个元素或一组元素。

版本新增：1.0

依赖：

- UI 核心（UI Core）
- 部件库（Widget Factory）
- 鼠标交互（Mouse Interaction）

注释：jQuery UI 可选择（Selectable）插件允许通过鼠标拖拽选择元素（有时被称为一个套索）。可以在按住 ctrl/meta 键的同时单击或拖动来选择多个（不连续的）元素。

附加说明：该部件要求一些功能性的 CSS，否则将无法工作。如果您创建了一个自定义的主题，请使用小部件指定的 CSS 文件作为起点。

快速导航

选项	
appendToautoRefreshcanceldelaydisableddistancefiltertolerance	destroydisab

appendTo

类型：Selector

描述：选择助手（套索）要被添加到哪一个元素。

代码实例：

初始化带有指定 `appendTo` 选项的 draggable：

```
$( ".selector" ).selectable({ appendTo: "#someElem" });
```

在初始化后，获取或设置 `appendTo` 选项：

```
// getter
var appendTo = $( ".selector" ).selectable( "option", "appendTo" );

// setter
$( ".selector" ).selectable( "option", "appendTo", "#someElem" );
```

默认值："body"

autoRefresh

类型：Boolean

描述：该选项决定是否在每个选择操作的开始时更新（重新计算）每个选择项的位置和尺寸。如果您有多个项目，您可能要设置该选项为 false，并手动调用 `[refresh()](#method-refresh)` 方法。

代码实例：

初始化带有指定 `autoRefresh` 选项的 draggable：

```
$( ".selector" ).selectable({ autoRefresh: false });
```

在初始化后，获取或设置 `autoRefresh` 选项：

```
// getter
var autoRefresh = $( ".selector" ).selectable( "option", "autoRefre

// setter
$( ".selector" ).selectable( "option", "autoRefresh", false );
```

默认值：true

cancel

类型：Selector

描述：防止从匹配选择器的元素上开始选择。

代码实例：

初始化带有指定 `cancel` 选项的 selectable：

```
$( ".selector" ).selectable({ cancel: "a,.cancel" });
```

在初始化后，获取或设置 `cancel` 选项：

```
// getter
var cancel = $( ".selector" ).selectable( "option", "cancel" );

// setter
$( ".selector" ).selectable( "option", "cancel", "a,.cancel" );
```

默认值："input, textarea, button, select, option"

delay

类型：Number

描述：鼠标按下后直到选择开始的时间，以毫秒计。该选项可以防止点击在某个元素上时不必要的选择。

代码实例：

初始化带有指定 `delay` 选项的 selectable：

```
$( ".selector" ).selectable({ delay: 150 });
```

在初始化后，获取或设置 `delay` 选项：

```
// getter
var delay = $( ".selector" ).selectable( "option", "delay" );

// setter
$( ".selector" ).selectable( "option", "delay", 150 );
```

默认值：0

disabled

类型：Boolean

描述：如果设置为 `true`，则禁用该 selectable。

代码实例：

初始化带有指定 `disabled` 选项的 selectable：

```
$( ".selector" ).selectable({ disabled: true });
```

在初始化后，获取或设置 `disabled` 选项：

```
// getter
var disabled = $( ".selector" ).selectable( "option", "disabled" );

// setter
$( ".selector" ).selectable( "option", "disabled", true );
```

默认值：false

distance

类型：Number

描述：鼠标按下后选择开始前必须移动的距离，以像素计。如果指定了该选项，选择只有在鼠标拖拽超出指定距离时才会开始。该选项可以防止点击在某个元素上时不必要的选择。

代码实例：

初始化带有指定 `distance` 选项的 selectable：

```
$( ".selector" ).selectable({ distance: 30 });
```

在初始化后，获取或设置 `distance` 选项：

```
// getter
var distance = $( ".selector" ).selectable( "option", "distance" );

// setter
$( ".selector" ).selectable( "option", "distance", 30 );
```

默认值：0

filter

类型：Selector

描述：要制作选择项（可被选择的）的匹配的子元素。

代码实例：

初始化带有指定 `filter` 选项的 selectable：

```
$( ".selector" ).selectable({ filter: "li" });
```

在初始化后，获取或设置 `filter` 选项：

```
// getter
var filter = $( ".selector" ).selectable( "option", "filter" );

// setter
$( ".selector" ).selectable( "option", "filter", "li" );
```

默认值："*"

tolerance

类型：String

描述：指定用于测试套索是否选择一个项目的模式。可能的值：

- `"fit"` ：套索完全重叠在项目上。
- `"touch"` ：套索重叠在项目上，任何比例皆可。

代码实例：

初始化带有指定 `tolerance` 选项的 selectable：

```
$( ".selector" ).selectable({ tolerance: "fit" });
```

在初始化后，获取或设置 `tolerance` 选项：

```
// getter
var tolerance = $( ".selector" ).selectable( "option", "tolerance"

// setter
$( ".selector" ).selectable( "option", "tolerance", "fit" );
```

默认值："touch"

方法

destroy()

类型：jQuery (plugin only)

描述：完全移除 selectable 功能。这会把元素返回到它的预初始化状态。

- 该方法不接受任何参数。

代码实例：

调用 destroy 方法：

```
$( ".selector" ).selectable( "destroy" );
```

disable()

类型：jQuery (plugin only)

描述：禁用 selectable。

- 该方法不接受任何参数。

代码实例：

调用 disable 方法：

```
$( ".selector" ).selectable( "disable" );
```

enable()

类型：jQuery (plugin only)

描述：启用 selectable。

- 该方法不接受任何参数。

代码实例：

调用 enable 方法：

```
$( ".selector" ).selectable( "enable" );
```

option(optionName)

类型：Object

描述：获取当前与指定的 `optionName` 关联的值。

- **optionName** 类型：String 描述：要获取的选项的名称。

代码实例：

调用该方法：

```
var isDisabled = $( ".selector" ).selectable( "option", "disabled"
```



option()

类型：PlainObject

描述：获取一个包含键/值对的对象，键/值对表示当前 selectable 选项哈希。

- 该方法不接受任何参数。

代码实例：

调用该方法：

```
var options = $( ".selector" ).selectable( "option" );
```

option(optionName, value)

类型：jQuery (plugin only)

描述：设置与指定的 `optionName` 关联的 selectable 选项的值。

- **optionName** 类型：String 描述：要设置的选项的名称。
- **value** 类型：Object 描述：要为选项设置的值。

代码实例：

调用该方法：

```
$( ".selector" ).selectable( "option", "disabled", true );
```

option(options)

类型：jQuery (plugin only)

描述：为 selectable 设置一个或多个选项。

- **options** 类型：Object 描述：要设置的 option-value 对。

代码实例：

调用该方法：

```
$( ".selector" ).selectable( "option", { disabled: true } );
```

refresh()

类型：jQuery (plugin only)

描述：更新每个选择项元素的位置和尺寸。当 `[autoRefresh](#option-autoRefresh)` 选项被设置为 false 时，该方法可用于手动重新计算每个选择项的位置和尺寸。

- 该方法不接受任何参数。

代码实例：

调用 refresh 方法：

```
$( ".selector" ).selectable( "refresh" );
```

widget()

类型：jQuery

描述：返回一个包含 selectable 元素的 `jQuery` 对象。

- 该方法不接受任何参数。

代码实例：

调用 widget 方法：

```
var widget = $( ".selector" ).selectable( "widget" );
```

事件

create(event, ui)

类型：selectablecreate

描述：当 selectable 被创建时触发。

- **event** 类型：Event
- **ui** 类型：Object

注意：`ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

代码实例：

初始化带有指定 create 回调的 selectable：

```
$( ".selector" ).selectable({
  create: function( event, ui ) {}
});
```

绑定一个事件监听器到 selectablecreate 事件：

```
$( ".selector" ).on( "selectablecreate", function( event, ui ) {} 
```

selected(event, ui)

类型：selectableselected

描述：当每个元素被添加选择时，在选择操作结尾触发。

- **event** 类型：Event
- **ui** 类型：Object
    - **selected** 类型：Element 描述：被选择的可选择项目。

代码实例：

初始化带有指定 selected 回调的 selectable：

```
$( ".selector" ).selectable({
  selected: function( event, ui ) {}
});
```

绑定一个事件监听器到 selectableselected 事件：

```
$( ".selector" ).on( "selectableselected", function( event, ui ) {
```

selecting(event, ui)

类型：selectableselecting

描述：当每个元素被添加选择时，在选择操作期间触发。

- **event** 类型：Event
- **ui** 类型：Object
    - **selecting** 类型：Element 描述：正被选择的当前可选择项目。

代码实例：

初始化带有指定 selecting 回调的 selectable：

```
$( ".selector" ).selectable({
  selecting: function( event, ui ) {}
});
```

绑定一个事件监听器到 selectableselecting 事件：

```
$( ".selector" ).on( "selectableselecting", function( event, ui )
```

start(event, ui)

类型：selectablestart

描述：在选择操作开头触发。

- **event** 类型：Event
- **ui** 类型：Object

注意：`ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

代码实例：

初始化带有指定 start 回调的 selectable：

```
$( ".selector" ).selectable({
  start: function( event, ui ) {}
});
```

绑定一个事件监听器到 selectablestart 事件：

```
$( ".selector" ).on( "selectablestart", function( event, ui ) {} )
```

stop(event, ui)

类型：selectablestop

描述：在选择操作结尾触发。

- **event** 类型：Event
- **ui** 类型：Object

注意：`ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

代码实例：

初始化带有指定 stop 回调的 selectable：

```
$( ".selector" ).selectable({
  stop: function( event, ui ) {}
});
```

绑定一个事件监听器到 selectablestop 事件：

```
$( ".selector" ).on( "selectablestop", function( event, ui ) {} );
```

unselected(event, ui)

类型：selectableunselected

描述：当每个元素从选择中被移除时，在选择操作结尾触发。

- **event** 类型：Event
- **ui** 类型：Object
  - **unselected** 类型：Element 描述：未被选择的可选择项目。

代码实例：

初始化带有指定 unselected 回调的 selectable：

```
$( ".selector" ).selectable({
  unselected: function( event, ui ) {}
});
```

绑定一个事件监听器到 selectableunselected 事件：

```
$( ".selector" ).on( "selectableunselected", function( event, ui )
```

unselecting(event, ui)

类型：selectableunselecting

描述：当每个元素从选择中被移除时，在选择操作期间触发。

- **event** 类型：Event
- **ui** 类型：Object
  - **unselecting** 类型：Element 描述：正未被选择的当前可选择项目。

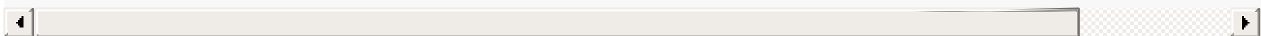
代码实例：

初始化带有指定 unselecting 回调的 selectable：

```
$( ".selector" ).selectable({
  unselecting: function( event, ui ) {}
});
```

绑定一个事件监听器到 selectableunselecting 事件：

```
$( ".selector" ).on( "selectableunselecting", function( event, ui )
```



实例

一个简单的 jQuery UI 可选择小部件（Selectable Widget）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>可选择小部件（Selectable Widget）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  #selectable .ui-selecting {
    background: #ccc;
  }
  #selectable .ui-selected {
    background: #999;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<ul id="selectable">
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
  <li>Item 4</li>
  <li>Item 5</li>
</ul>

<script>
$( "#selectable" ).selectable();
</script>

</body>
</html>
```

jQuery UI API - 可排序小部件（Sortable Widget）

所属类别

交互（Interactions）

用法

描述：使用鼠标调整列表中或者网格中元素的排序。

版本新增：1.0

依赖：

- UI 核心（UI Core）
- 部件库（Widget Factory）
- 鼠标交互（Mouse Interaction）

注释：jQuery UI 可排序（Sortable）插件让被选元素通过鼠标拖拽进行排序。

注释：为了排序表格行，`tbody` 必须是可排序的（sortable），而不是 `table`。

快速导航

appendToaxiscancelconnectWithcontainmentcursorcursorAtdelaydisableddistanc

appendTo

类型：jQuery 或 Element 或 Selector 或 String

描述：当拖拽时，通过鼠标移动的助手被追加到哪里（例如，解决 overlap/zIndex 问题）。

支持多个类型：

- **jQuery**：一个 jQuery 对象，包含助手（helper）要追加到的元素。
- **Element**：要追加助手（helper）的元素。
- **Selector**：一个选择器，指定哪个元素要追加助手（helper）。
- **String**：字符串 `"parent"` 将促使助手（helper）成为 sortable 项目的同级。

代码实例：

初始化带有指定 `appendTo` 选项的 sortable：

```
$( ".selector" ).sortable({ appendTo: document.body });
```

在初始化后，获取或设置 `appendTo` 选项：

```
// getter
var appendTo = $( ".selector" ).sortable( "option", "appendTo" );

// setter
$( ".selector" ).sortable( "option", "appendTo", document.body );
```

默认值："parent"

axis

类型：String

描述：如果定义了该选项，项目只能在水平或垂直方向上被拖拽。可能的值： `"x"` ， `"y"` 。

代码实例：

初始化带有指定 `axis` 选项的 sortable：

```
$( ".selector" ).sortable({ axis: "x" });
```

在初始化后，获取或设置 `axis` 选项：

```
// getter
var axis = $( ".selector" ).sortable( "option", "axis" );

// setter
$( ".selector" ).sortable( "option", "axis", "x" );
```

默认值：false

cancel

类型：Selector

描述：防止从匹配选择器的元素上开始排序。

代码实例：

初始化带有指定 `cancel` 选项的 sortable：

```
$( ".selector" ).sortable({ cancel: "a,button" });
```

在初始化后，获取或设置 `cancel` 选项：

```
// getter
var cancel = $( ".selector" ).sortable( "option", "cancel" );

// setter
$( ".selector" ).sortable( "option", "cancel", "a,button" );
```

默认值："input, textarea, button, select, option"

connectWith

类型：Selector

描述：列表中的项目需被连接的其他 sortable 元素的选择器。这是一个单向关系，如果您想要项目被双向连接，必须在两个 sortable 元素上都设置 `connectWith` 选项。

代码实例：

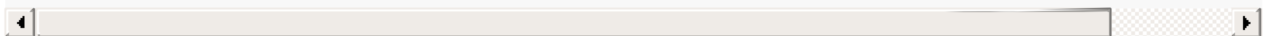
初始化带有指定 `connectWith` 选项的 sortable：

```
$( ".selector" ).sortable({ connectWith: "#shopping-cart" });
```

在初始化后，获取或设置 `connectWith` 选项：

```
// getter
var connectWith = $( ".selector" ).sortable( "option", "connectWit

// setter
$( ".selector" ).sortable( "option", "connectWith", "#shopping-car
```



默认值：false

containment

类型：Element 或 Selector 或 String

描述：定义拖拽时，sortable 项目被约束的边界。

注释：为 containment 指定的元素必须有一个已计算的宽度和高度（尽管它不需要显式）。例如，如果您有 `float: left` sortable 子元素，并指定了 `containment: "parent"` ，请确保在 sortable/parent 容器上有 `float: left` ，否则它将有 `height: 0` ，导致未定义的行为。

支持多个类型：

- **Element**：一个要作为容器使用的元素。
- **Selector**：一个选择器，指定一个要作为容器使用的元素。
- **String**：一个字符串，标识一个要作为容器使用的元素。可能的值： `"parent"` 、 `"document"` 、 `"window"` 。

代码实例：

初始化带有指定 `containment` 选项的 sortable：

```
$( ".selector" ).sortable({ containment: "parent" });
```

在初始化后，获取或设置 `containment` 选项：

```
// getter
var containment = $( ".selector" ).sortable( "option", "containment

// setter
$( ".selector" ).sortable( "option", "containment", "parent" );
```

默认值：false

cursor

类型：String

描述：定义当排序时被显示的光标。

代码实例：

初始化带有指定 `cursor` 选项的 sortable：

```
$( ".selector" ).sortable({ cursor: "move" });
```

在初始化后，获取或设置 `cursor` 选项：

```
// getter
var cursor = $( ".selector" ).sortable( "option", "cursor" );

// setter
$( ".selector" ).sortable( "option", "cursor", "move" );
```

默认值："auto"

cursorAt

类型：Object

描述：移动排序元素或助手（helper），这样光标总是出现，以便从相同的位置进行拖拽。坐标可通过一个或两个键的组合成一个哈希给出：`{ top, left, right, bottom }`。

代码实例：

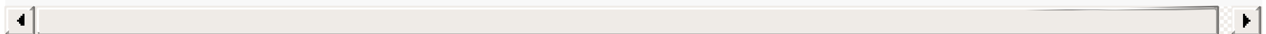
初始化带有指定 `cursorAt` 选项的 sortable：

```
$( ".selector" ).sortable({ cursorAt: { left: 5 } });
```

在初始化后，获取或设置 `cursorAt` 选项：

```
// getter
var cursorAt = $( ".selector" ).sortable( "option", "cursorAt" );

// setter
$( ".selector" ).sortable( "option", "cursorAt", { left: 5 } );
```



默认值：false

delay

类型：Integer

描述：鼠标按下后直到排序开始的时间，以毫秒计。该选项可以防止点击在某个元素上时不必要的拖拽。

代码实例：

初始化带有指定 `delay` 选项的 sortable：

```
$( ".selector" ).sortable({ delay: 150 });
```

在初始化后，获取或设置 `delay` 选项：

```
// getter
var delay = $( ".selector" ).sortable( "option", "delay" );

// setter
$( ".selector" ).sortable( "option", "delay", 150 );
```

默认值：0

disabled

类型：Boolean

描述：如果设置为 `true`，则禁用该 sortable。

代码实例：

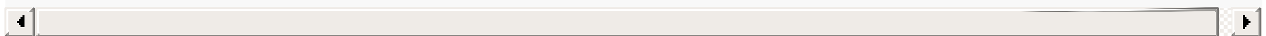
初始化带有指定 `disabled` 选项的 sortable：

```
$( ".selector" ).sortable({ disabled: true });
```

在初始化后，获取或设置 `disabled` 选项：

```
// getter
var disabled = $( ".selector" ).sortable( "option", "disabled" );

// setter
$( ".selector" ).sortable( "option", "disabled", true );
```



默认值：false

distance

类型：Number

描述：鼠标按下后排序开始前必须移动的距离，以像素计。如果指定了该选项，排序只有在鼠标拖拽超出指定距离时才会开始。该选项可以用于允许在一个手柄内的元素上点击。

代码实例：

初始化带有指定 `distance` 选项的 sortable：

```
$( ".selector" ).sortable({ distance: 5 });
```

在初始化后，获取或设置 `distance` 选项：

```
// getter
var distance = $( ".selector" ).sortable( "option", "distance" );

// setter
$( ".selector" ).sortable( "option", "distance", 5 );
```

默认值：1

dropOnEmpty

类型：Boolean

描述：如果为 `false`，从该 sortable 的项目不能被放置在空连接的 sortable 上（请查看 `[connectWith](#option-connectWith)` 选项）。

代码实例：

初始化带有指定 `dropOnEmpty` 选项的 sortable：

```
$( ".selector" ).sortable({ dropOnEmpty: false });
```

在初始化后，获取或设置 `dropOnEmpty` 选项：

```
// getter
var dropOnEmpty = $( ".selector" ).sortable( "option", "dropOnEmpty

// setter
$( ".selector" ).sortable( "option", "dropOnEmpty", false );
```

默认值：true

forceHelperSize

类型：Boolean

描述：如果为 `true`，强制助手（helper）有一个尺寸。

代码实例：

初始化带有指定 `forceHelperSize` 选项的 sortable：

```
$( ".selector" ).sortable({ forceHelperSize: true });
```

在初始化后，获取或设置 `forceHelperSize` 选项：

```
// getter
var forceHelperSize = $( ".selector" ).sortable( "option", "forceHe

// setter
$( ".selector" ).sortable( "option", "forceHelperSize", true );
```

默认值：false

forcePlaceholderSize

类型：Boolean

描述：如果为 `true`，强制占位符（placeholder）有一个尺寸。

代码实例：

初始化带有指定 `forcePlaceholderSize` 选项的 sortable：

```
$( ".selector" ).sortable({ forcePlaceholderSize: true });
```

在初始化后，获取或设置 `forcePlaceholderSize` 选项：

```
// getter
var forcePlaceholderSize = $( ".selector" ).sortable( "option", "fo

// setter
$( ".selector" ).sortable( "option", "forcePlaceholderSize", true )
```

默认值：false

grid

类型：Array

描述：对齐排序元素或助手（helper）到网格，每个 x 和 y 像素。数组形式必须是 `[ x, y ]`。

代码实例：

初始化带有指定 `grid` 选项的 sortable：

```
$( ".selector" ).sortable({ grid: [ 20, 10 ] });
```

在初始化后，获取或设置 `grid` 选项：

```
// getter
var grid = $( ".selector" ).sortable( "option", "grid" );

// setter
$( ".selector" ).sortable( "option", "grid", [ 20, 10 ] );
```

默认值：false

handle

类型：Selector 或 Element

描述：如果指定了该选项，则限制在指定的元素上开始排序。

代码实例：

初始化带有指定 `handle` 选项的 sortable：

```
$( ".selector" ).sortable({ handle: ".handle" });
```

在初始化后，获取或设置 `handle` 选项：

```
// getter
var handle = $( ".selector" ).sortable( "option", "handle" );

// setter
$( ".selector" ).sortable( "option", "handle", ".handle" );
```

默认值：false

helper

类型：String 或 Function()

描述：允许一个 helper 元素用于拖拽显示。

支持多个类型：

- **String**：如果设置为 `"clone"`，元素将被克隆，且克隆将被拖拽。
- **Function**：一个函数，将返回拖拽时要使用的 DOMElement。函数接收事件，且元素正被排序。

代码实例：

初始化带有指定 `helper` 选项的 sortable：

```
$( ".selector" ).sortable({ helper: "clone" });
```

在初始化后，获取或设置 `helper` 选项：

```
// getter
var helper = $( ".selector" ).sortable( "option", "helper" );

// setter
$( ".selector" ).sortable( "option", "helper", "clone" );
```

默认值："original"

items

类型：Selector

描述：指定元素内的哪一个项目应是 sortable。

代码实例：

初始化带有指定 `items` 选项的 sortable：

```
$( ".selector" ).sortable({ items: "&gt; li" });
```

在初始化后，获取或设置 `items` 选项：

```
// getter
var items = $( ".selector" ).sortable( "option", "items" );

// setter
$( ".selector" ).sortable( "option", "items", "&gt; li" );
```

默认值："> *"

opacity

类型：Number

描述：当排序时助手（helper）的不透明度。从 `0.01` 到 `1` 。

代码实例：

初始化带有指定 `opacity` 选项的 sortable：

```
$( ".selector" ).sortable({ opacity: 0.5 });
```

在初始化后，获取或设置 `opacity` 选项：

```
// getter
var opacity = $( ".selector" ).sortable( "option", "opacity" );

// setter
$( ".selector" ).sortable( "option", "opacity", 0.5 );
```

默认值：false

placeholder

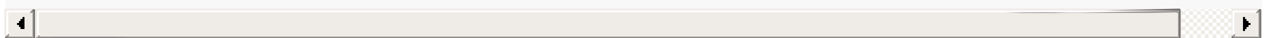
类型：String

描述：要应用的 class 名称，否则为白色空白。

代码实例：

初始化带有指定 `placeholder` 选项的 sortable：

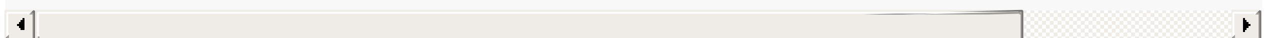
```
$( ".selector" ).sortable({ placeholder: "sortable-placeholder" });
```



在初始化后，获取或设置 `placeholder` 选项：

```
// getter
var placeholder = $( ".selector" ).sortable( "option", "placeholde

// setter
$( ".selector" ).sortable( "option", "placeholder", "sortable-place
```



默认值：false

revert

类型：Boolean 或 Number

描述：sortable 项目是否使用一个流畅的动画还原到它的新位置。

支持多个类型：

- **Boolean**：当设置为 `true`，该项目将会使用动画，动画使用默认的持续时间。
- **Number**：动画的持续时间，以毫秒计。

代码实例：

初始化带有指定 `revert` 选项的 sortable：

```
$( ".selector" ).sortable({ revert: true });
```

在初始化后，获取或设置 `revert` 选项：

```
// getter
var revert = $( ".selector" ).sortable( "option", "revert" );

// setter
$( ".selector" ).sortable( "option", "revert", true );
```

默认值：false

scroll

类型：Boolean

描述：如果设置为 `true`，当到达边缘时页面会滚动。

代码实例：

初始化带有指定 `scroll` 选项的 sortable：

```
$( ".selector" ).sortable({ scroll: false });
```

在初始化后，获取或设置 `scroll` 选项：

```
// getter
var scroll = $( ".selector" ).sortable( "option", "scroll" );

// setter
$( ".selector" ).sortable( "option", "scroll", false );
```

默认值：true

scrollSensitivity

类型：Number

描述：定义鼠标距离边缘多少距离时开始滚动。

代码实例：

初始化带有指定 `scrollSensitivity` 选项的 sortable：

```
$( ".selector" ).sortable({ scrollSensitivity: 10 });
```

在初始化后，获取或设置 `scrollSensitivity` 选项：

```
// getter
var scrollSensitivity = $( ".selector" ).sortable( "option", "scrol

// setter
$( ".selector" ).sortable( "option", "scrollSensitivity", 10 );
```

默认值：20

scrollSpeed

类型：Number

描述：当鼠标指针获取到在 `[scrollSensitivity](#option-scrollSensitivity)` 距离内时，窗体滚动的速度。如果 `scroll` 选项是 `false` 则忽略。

代码实例：

初始化带有指定 `scrollSpeed` 选项的 sortable：

```
$( ".selector" ).sortable({ scrollSpeed: 40 });
```

在初始化后，获取或设置 `scrollSpeed` 选项：

```
// getter
var scrollSpeed = $( ".selector" ).sortable( "option", "scrollSpeed

// setter
$( ".selector" ).sortable( "option", "scrollSpeed", 40 );
```

默认值：20

tolerance

类型：String

描述：指定用于测试项目被移动时是否覆盖在另一个项目上的模式。可能的值：

- `"intersect"`：项目至少 50% 重叠在其他项目上。
- `"pointer"`：鼠标指针重叠在其他项目上。

代码实例：

初始化带有指定 `tolerance` 选项的 sortable：

```
$( ".selector" ).sortable({ tolerance: "pointer" });
```

在初始化后，获取或设置 `tolerance` 选项：

```
// getter
var tolerance = $( ".selector" ).sortable( "option", "tolerance" );

// setter
$( ".selector" ).sortable( "option", "tolerance", "pointer" );
```

默认值："intersect"

zIndex

类型：Integer

描述：当被排序时，元素/助手（helper）的 Z-index。

代码实例：

初始化带有指定 `zIndex` 选项的 sortable：

```
$( ".selector" ).sortable({ zIndex: 9999 });
```

在初始化后，获取或设置 `zIndex` 选项：

```
// getter
var zIndex = $( ".selector" ).sortable( "option", "zIndex" );

// setter
$( ".selector" ).sortable( "option", "zIndex", 9999 );
```

默认值：1000

方法

cancel()

类型：jQuery (plugin only)

描述：当前排序开始时，取消一个在当前 sortable 中的改变，且恢复到之前的状态。在 stop 和 receive 回调函数中非常有用。

- 该方法不接受任何参数。

代码实例：

调用 cancel 方法：

```
$( ".selector" ).sortable( "cancel" );
```

destroy()

类型：jQuery (plugin only)

描述：完全移除 sortable 功能。这会把元素返回到它的预初始化状态。

- 该方法不接受任何参数。

代码实例：

调用 destroy 方法：

```
$( ".selector" ).sortable( "destroy" );
```

disable()

类型：jQuery (plugin only)

描述：禁用 sortable。

- 该方法不接受任何参数。

代码实例：

调用 disable 方法：

```
$( ".selector" ).sortable( "disable" );
```

enable()

类型：jQuery (plugin only)

描述：启用 sortable。

- 该方法不接受任何参数。

代码实例：

调用 enable 方法：

```
$( ".selector" ).sortable( "enable" );
```

option(optionName)

类型：Object

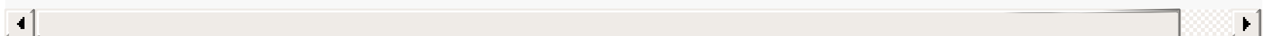
描述：获取当前与指定的 `optionName` 关联的值。

- **optionName** 类型：String 描述：要获取的选项的名称。

代码实例：

调用该方法：

```
var isDisabled = $( ".selector" ).sortable( "option", "disabled" );
```



option()

类型：PlainObject

描述：获取一个包含键/值对的对象，键/值对表示当前 draggable 选项哈希。

- 该方法不接受任何参数。

代码实例：

调用该方法：

```
var options = $( ".selector" ).sortable( "option" );
```

option(optionName, value)

类型：jQuery (plugin only)

描述：设置与指定的 `optionName` 关联的 sortable 选项的值。

- **optionName** 类型：String 描述：要设置的选项的名称。
- **value** 类型：Object 描述：要为选项设置的值。

代码实例：

调用该方法：

```
$( ".selector" ).sortable( "option", "disabled", true );
```

option(options)

类型：jQuery (plugin only)

描述：为 sortable 设置一个或多个选项。

- **options** 类型：Object 描述：要设置的 option-value 对。

代码实例：

调用该方法：

```
$( ".selector" ).sortable( "option", { disabled: true } );
```

refresh()

类型：jQuery (plugin only)

描述：刷新 sortable 项目。触发所有 sortable 项目重新加载，导致新的项目被认可。

- 该方法不接受任何参数。

代码实例：

调用 refresh 方法：

```
$( ".selector" ).sortable( "refresh" );
```

refreshPositions()

类型：jQuery (plugin only)

描述：刷新 sortable 项目的缓存位置。调用该方法刷新所有 sortable 的已缓存的项目位置。

- 该方法不接受任何参数。

代码实例：

调用 refreshPositions 方法：

```
$( ".selector" ).sortable( "refreshPositions" );
```

serialize(options)

类型：String

描述：序列化 sortable 的项目 `id` 为一个 form/ajax 可提交的字符串。调用该方法会产生一个可被追加到任何 url 中的哈希，以便简单地把一个新的项目顺序提交回服务器。

默认情况下，它通过每个项目的 `id` 进行工作，id 格式为 `"setname_number"`，且它会产生一个形如 `"setname[]=number&setname[]=number"` 的哈希。

注释：如果序列化返回一个空的字符串，请确认 `id` 属性包含一个下划线（）。形式必须是 *`"set_number"`*。例如，一个 *`id`* 属性为 *`"foo_1"`*、*`"foo_5"`*、*`"foo_2"`* 的 *3* 元素列表将序列化为 *`"foo[]=1&foo[]=5&foo[]=2"`*。您可以使用下划线（）、等号（=）或连字符（-）来分隔集合和数字。例如，`"foo=1"`、`"foo-1"`、`"foo_1"` 所有都序列化为 `"foo[]=1"`。

- **options** 类型：Object 要自定义序列化的选项。
  - **key**（默认值：`属性中分隔符前面的部分`） 类型：String 描述：把 `part1[]` 替换为指定的值。
  - **attribute**（默认值：`"id"`） 类型：String 描述：值要使用的属性名称。
  - **expression**（默认值：`/(.+)[-=_](.+)/`） 类型：RegExp 描述：用于把属性值分隔为键和值两部分的正则表达式。

代码实例：

调用 serialize 方法：

```
var sorted = $( ".selector" ).sortable( "serialize", { key: "sort"
```

toArray(options)

类型：Array

描述：序列化 sortable 的项目 `id` 为一个字符串的数组。

- **options** 类型：Object 要自定义序列化的选项。
  - **attribute**（默认值：`"id"`） 类型：String 描述：值要使用的属性名称。

代码实例：

调用 toArray 方法：

```
var sortedIDs = $( ".selector" ).sortable( "toArray" );
```

widget()

类型：jQuery

描述：返回一个包含 sortable 元素的 `jQuery` 对象。

- 该方法不接受任何参数。

代码实例：

调用 widget 方法：

```
var widget = $( ".selector" ).sortable( "widget" );
```

事件

activate(event, ui)

类型：sortactivate

描述：当使用被连接列表时触发该事件，每个被连接列表在拖拽开始时接收它。

- **event** 类型：Event
- **ui** 类型：Object
  - **helper** 类型：jQuery 描述：jQuery 对象，表示被排序的助手（helper）。
  - **item** 类型：jQuery 描述：jQuery 对象，表示当前被拖拽的元素。
  - **offset** 类型：Object 描述：助手（helper）的当前的绝对位置，表示为 `{ top, left }`。
  - **position** 类型：Object 描述：助手（helper）的当前位置，表示为 `{ top, left }`。
  - **originalPosition** 类型：Object 描述：元素的原始位置，表示为

  `{ top, left }` 。
  - **sender** 类型：jQuery 描述：如果从一个 sortable 移动到另一个 sortable，项目来自的那个 sortable。
  - **placeholder** 类型：jQuery 描述：jQuery 对象，表示被作为占位符使用的元素。

代码实例：

初始化带有指定 activate 回调的 sortable：

```
$( ".selector" ).sortable({
activate: function( event, ui ) {}
});
```

绑定一个事件监听器到 sortactivate 事件：

```
$( ".selector" ).on( "sortactivate", function( event, ui ) {} );
```

beforeStop(event, ui)

类型：sortbeforestop

描述：当排序停止时触发该事件，除了当占位符（placeholder）/助手（helper）仍然可用时。

- **event** 类型：Event
- **ui** 类型：Object
  - **helper** 类型：jQuery 描述：jQuery 对象，表示被排序的助手（helper）。
  - **item** 类型：jQuery 描述：jQuery 对象，表示当前被拖拽的元素。
  - **offset** 类型：Object 描述：助手（helper）的当前的绝对位置，表示为 `{ top, left }` 。
  - **position** 类型：Object 描述：助手（helper）的当前位置，表示为 `{ top, left }` 。
  - **originalPosition** 类型：Object 描述：元素的原始位置，表示为 `{ top, left }` 。
  - **sender** 类型：jQuery 描述：如果从一个 sortable 移动到另一个 sortable，项目来自的那个 sortable。
  - **placeholder** 类型：jQuery 描述：jQuery 对象，表示被作为占位符使用的元素。

代码实例：

初始化带有指定 beforeStop 回调的 sortable：

```
$( ".selector" ).sortable({
beforeStop: function( event, ui ) {}
});
```

绑定一个事件监听器到 sortbeforestop 事件：

```
$( ".selector" ).on( "sortbeforestop", function( event, ui ) {} );
```

change(event, ui)

类型：sortchange

描述：在排序期间触发该事件，除了当 DOM 位置改变时。

- **event** 类型：Event
- **ui** 类型：Object
    - **helper** 类型：jQuery 描述：jQuery 对象，表示被排序的助手（helper）。
    - **item** 类型：jQuery 描述：jQuery 对象，表示当前被拖拽的元素。
    - **offset** 类型：Object 描述：助手（helper）的当前的绝对位置，表示为 `{ top, left }` 。
    - **position** 类型：Object 描述：助手（helper）的当前位置，表示为 `{ top, left }` 。
    - **originalPosition** 类型：Object 描述：元素的原始位置，表示为 `{ top, left }` 。
    - **sender** 类型：jQuery 描述：如果从一个 sortable 移动到另一个 sortable，项目来自的那个 sortable。
    - **placeholder** 类型：jQuery 描述：jQuery 对象，表示被作为占位符使用的元素。

代码实例：

初始化带有指定 change 回调的 sortable：

```
$( ".selector" ).sortable({
change: function( event, ui ) {}
});
```

绑定一个事件监听器到 sortchange 事件：

```
$( ".selector" ).on( "sortchange", function( event, ui ) {} );
```

create(event, ui)

类型：sortcreate

描述：当 droppable 被创建时触发。

- **event** 类型：Event
- **ui** 类型：Object

注意：`ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

代码实例：

初始化带有指定 create 回调的 sortable：

```
$( ".selector" ).sortable({
create: function( event, ui ) {}
});
```

绑定一个事件监听器到 sortcreate 事件：

```
$( ".selector" ).on( "sortcreate", function( event, ui ) {} );
```

deactivate(event, ui)

类型：sortdeactivate

描述：当排序停止时触发该事件，该事件传播到所有可能的连接列表。。

- **event** 类型：Event
- **ui** 类型：Object
  - **helper** 类型：jQuery 描述：jQuery 对象，表示被排序的助手（helper）。
  - **item** 类型：jQuery 描述：jQuery 对象，表示当前被拖拽的元素。
  - **offset** 类型：Object 描述：助手（helper）的当前的绝对位置，表示为 `{ top, left }`。
  - **position** 类型：Object 描述：助手（helper）的当前位置，表示为 `{ top, left }`。
  - **originalPosition** 类型：Object 描述：元素的原始位置，表示为 `{ top, left }`。
  - **sender** 类型：jQuery 描述：如果从一个 sortable 移动到另一个 sortable，项目来自的那个 sortable。
  - **placeholder** 类型：jQuery 描述：jQuery 对象，表示被作为占位符使用的元素。

代码实例：

初始化带有指定 deactivate 回调的 sortable：

```
$( ".selector" ).sortable({
deactivate: function( event, ui ) {}
});
```

绑定一个事件监听器到 sortdeactivate 事件：

```
$( ".selector" ).on( "sortdeactivate", function( event, ui ) {} );
```

out(event, ui)

类型：sortout

描述：当一个 sortable 项目从一个 sortable 列表移除时触发该事件。

注释：当一个 sortable 项目被撤销时也会触发该事件。

- **event** 类型：Event
- **ui** 类型：Object
  - **helper** 类型：jQuery 描述：jQuery 对象，表示被排序的助手（helper）。
  - **item** 类型：jQuery 描述：jQuery 对象，表示当前被拖拽的元素。
  - **offset** 类型：Object 描述：助手（helper）的当前的绝对位置，表示为 `{ top, left }` 。
  - **position** 类型：Object 描述：助手（helper）的当前位置，表示为 `{ top, left }` 。
  - **originalPosition** 类型：Object 描述：元素的原始位置，表示为 `{ top, left }` 。
  - **sender** 类型：jQuery 描述：如果从一个 sortable 移动到另一个 sortable，项目来自的那个 sortable。
  - **placeholder** 类型：jQuery 描述：jQuery 对象，表示被作为占位符使用的元素。

代码实例：

初始化带有指定 out 回调的 sortable：

```
$( ".selector" ).sortable({
out: function( event, ui ) {}
});
```

绑定一个事件监听器到 sortout 事件：

```
$( ".selector" ).on( "sortout", function( event, ui ) {} );
```

over(event, ui)

类型：sortover

描述：当一个 sortable 项目移动到一个 sortable 列表时触发该事件。

- **event** 类型：Event
- **ui** 类型：Object
    - **helper** 类型：jQuery 描述：jQuery 对象，表示被排序的助手（helper）。
    - **item** 类型：jQuery 描述：jQuery 对象，表示当前被拖拽的元素。
    - **offset** 类型：Object 描述：助手（helper）的当前的绝对位置，表示为 `{ top, left }` 。
    - **position** 类型：Object 描述：助手（helper）的当前位置，表示为 `{ top, left }` 。
    - **originalPosition** 类型：Object 描述：元素的原始位置，表示为 `{ top, left }` 。
    - **sender** 类型：jQuery 描述：如果从一个 sortable 移动到另一个 sortable，项目来自的那个 sortable。
    - **placeholder** 类型：jQuery 描述：jQuery 对象，表示被作为占位符使用的元素。

代码实例：

初始化带有指定 over 回调的 sortable：

```
$( ".selector" ).sortable({
out: function( event, ui ) {}
});
```

绑定一个事件监听器到 sortover 事件：

```
$( ".selector" ).on( "sortout", function( event, ui ) {} );
```

receive(event, ui)

类型：sortreceive

描述：当来自一个连接的 sortable 列表的一个项目被放置到另一个列表时触发该事件。后者是事件目标。

- **event** 类型：Event
- **ui** 类型：Object
    - **helper** 类型：jQuery 描述：jQuery 对象，表示被排序的助手（helper）。
    - **item** 类型：jQuery 描述：jQuery 对象，表示当前被拖拽的元素。
    - **offset** 类型：Object 描述：助手（helper）的当前的绝对位置，表示为

`{ top, left }`。
    - **position** 类型：Object 描述：助手（helper）的当前位置，表示为 `{ top, left }`。
    - **originalPosition** 类型：Object 描述：元素的原始位置，表示为 `{ top, left }`。
    - **sender** 类型：jQuery 描述：如果从一个 sortable 移动到另一个 sortable，项目来自的那个 sortable。
    - **placeholder** 类型：jQuery 描述：jQuery 对象，表示被作为占位符使用的元素。

代码实例：

初始化带有指定 receive 回调的 sortable：

```
$( ".selector" ).sortable({
receive: function( event, ui ) {}
});
```

绑定一个事件监听器到 sortreceive 事件：

```
$( ".selector" ).on( "sortreceive", function( event, ui ) {} );
```

remove(event, ui)

类型：sortremove

描述：当来自一个连接的 sortable 列表的一个项目被放置到另一个列表时触发该事件。前者是事件目标。

- **event** 类型：Event
- **ui** 类型：Object
    - **helper** 类型：jQuery 描述：jQuery 对象，表示被排序的助手（helper）。
    - **item** 类型：jQuery 描述：jQuery 对象，表示当前被拖拽的元素。
    - **offset** 类型：Object 描述：助手（helper）的当前的绝对位置，表示为 `{ top, left }`。
    - **position** 类型：Object 描述：助手（helper）的当前位置，表示为 `{ top, left }`。
    - **originalPosition** 类型：Object 描述：元素的原始位置，表示为 `{ top, left }`。
    - **placeholder** 类型：jQuery 描述：jQuery 对象，表示被作为占位符使用的元素。

代码实例：

初始化带有指定 remove 回调的 sortable：

```
$( ".selector" ).sortable({
remove: function( event, ui ) {}
});
```

绑定一个事件监听器到 sortremove 事件：

```
$( ".selector" ).on( "sortremove", function( event, ui ) {} );
```

sort(event, ui)

类型：sort

描述：在排序期间触发该事件。

- **event** 类型：Event
- **ui** 类型：Object
  - **helper** 类型：jQuery 描述：jQuery 对象，表示被排序的助手（helper）。
  - **item** 类型：jQuery 描述：jQuery 对象，表示当前被拖拽的元素。
  - **offset** 类型：Object 描述：助手（helper）的当前的绝对位置，表示为 `{ top, left }` 。
  - **position** 类型：Object 描述：助手（helper）的当前位置，表示为 `{ top, left }` 。
  - **originalPosition** 类型：Object 描述：元素的原始位置，表示为 `{ top, left }` 。
  - **sender** 类型：jQuery 描述：如果从一个 sortable 移动到另一个 sortable，项目来自的那个 sortable。
  - **placeholder** 类型：jQuery 描述：jQuery 对象，表示被作为占位符使用的元素。

代码实例：

初始化带有指定 sort 回调的 sortable：

```
$( ".selector" ).sortable({
sort: function( event, ui ) {}
});
```

绑定一个事件监听器到 sort 事件：

```
$( ".selector" ).on( "sort", function( event, ui ) {} );
```

start(event, ui)

类型：sortstart

描述：当排序开始时触发该事件。

- **event** 类型：Event
- **ui** 类型：Object
    - **helper** 类型：jQuery 描述：jQuery 对象，表示被排序的助手（helper）。
    - **item** 类型：jQuery 描述：jQuery 对象，表示当前被拖拽的元素。
    - **offset** 类型：Object 描述：助手（helper）的当前的绝对位置，表示为 `{ top, left }` 。
    - **position** 类型：Object 描述：助手（helper）的当前位置，表示为 `{ top, left }` 。
    - **originalPosition** 类型：Object 描述：元素的原始位置，表示为 `{ top, left }` 。
    - **sender** 类型：jQuery 描述：如果从一个 sortable 移动到另一个 sortable，项目来自的那个 sortable。
    - **placeholder** 类型：jQuery 描述：jQuery 对象，表示被作为占位符使用的元素。

代码实例：

初始化带有指定 start 回调的 sortable：

```
$( ".selector" ).sortable({
start: function( event, ui ) {}
});
```

绑定一个事件监听器到 sortstart 事件：

```
$( ".selector" ).on( "sortstart", function( event, ui ) {} );
```

stop(event, ui)

类型：sortstop

描述：当排序停止时触发该事件。

- **event** 类型：Event
- **ui** 类型：Object
    - **helper** 类型：jQuery 描述：jQuery 对象，表示被排序的助手（helper）。
    - **item** 类型：jQuery 描述：jQuery 对象，表示当前被拖拽的元素。
    - **offset** 类型：Object 描述：助手（helper）的当前的绝对位置，表示为 `{ top, left }` 。
    - **position** 类型：Object 描述：助手（helper）的当前位置，表示为 `{ top, left }` 。

  - **originalPosition** 类型：Object 描述：元素的原始位置，表示为 `{ top, left }` 。
  - **sender** 类型：jQuery 描述：如果从一个 sortable 移动到另一个 sortable，项目来自的那个 sortable。
  - **placeholder** 类型：jQuery 描述：jQuery 对象，表示被作为占位符使用的元素。

代码实例：

初始化带有指定 stop 回调的 sortable：

```
$( ".selector" ).sortable({
stop: function( event, ui ) {}
});
```

绑定一个事件监听器到 sortstop 事件：

```
$( ".selector" ).on( "sortstop", function( event, ui ) {} );
```

update(event, ui)

类型：sortupdate

描述：当用户停止排序且 DOM 位置改变时触发该事件。

- **event** 类型：Event
- **ui** 类型：Object
  - **helper** 类型：jQuery 描述：jQuery 对象，表示被排序的助手（helper）。
  - **item** 类型：jQuery 描述：jQuery 对象，表示当前被拖拽的元素。
  - **offset** 类型：Object 描述：助手（helper）的当前的绝对位置，表示为 `{ top, left }` 。
  - **position** 类型：Object 描述：助手（helper）的当前位置，表示为 `{ top, left }` 。
  - **originalPosition** 类型：Object 描述：元素的原始位置，表示为 `{ top, left }` 。
  - **sender** 类型：jQuery 描述：如果从一个 sortable 移动到另一个 sortable，项目来自的那个 sortable。
  - **placeholder** 类型：jQuery 描述：jQuery 对象，表示被作为占位符使用的元素。

代码实例：

初始化带有指定 update 回调的 sortable：

```
$( ".selector" ).sortable({
update: function( event, ui ) {}
});
```

绑定一个事件监听器到 sortupdate 事件：

```
$( ".selector" ).on( "sortupdate", function( event, ui ) {} );
```

实例

一个简单的 jQuery UI 可排序小部件（Sortable Widget）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>可排序小部件（Sortable Widget）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<ul id="sortable">
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
  <li>Item 4</li>
  <li>Item 5</li>
</ul>

<script>$("#sortable").sortable();</script>

</body>
</html>
```

jQuery UI API 类别 - 方法重载（Method Overrides）

jQuery UI 重载了几个内置的 jQuery 方法，以提供额外的功能。当使用这些重载时，确保加载 jQuery UI 是很重要的。如果 jQuery UI 未加载，方法依然存在，但预期的功能将不可用，这会导致难以追踪的错误。

API	描述	也属于类别
.addClass()	当动画样式改变时，为匹配的元素集合内的每个元素添加指定的 Class。	特效（Effects） | 特效核心（Effects Core）
.focus()	异步聚焦到一个元素。	方法（Method） | UI 核心（UI Core）
.hide()	使用自定义效果来隐藏匹配的元素。	特效（Effects） | 特效核心（Effects Core） | 方法（Method）
.position()	相对另一个元素定位一个元素。	方法（Method） | 实用工具（Utilities）
.removeClass()	当动画样式改变时，为匹配的元素集合内的每个元素移除指定的 Class。	特效（Effects） | 特效核心（Effects Core）
.show()	使用自定义效果来显示匹配的元素。	特效（Effects） | 特效核心（Effects Core） | 方法（Method）
.toggle()	使用自定义效果来显示或隐藏匹配的元素。	特效（Effects） | 特效核心（Effects Core） | 方法（Method）
.toggleClass()	当动画样式改变时，根据 Class 是否存在以及 switch 参数的值，为匹配的元素集合内的每个元素添加或移除一个或多个 Class。	特效（Effects） | 特效核心（Effects Core）

jQuery UI API - .focus()

所属类别

方法重载（Method Overrides） | 方法（Methods） | UI 核心（UI Core）

用法

描述：异步聚焦到一个元素。

返回：jQuery

```
.focus( delay [, callback ] )Returns: jQuery
```

参数	类型	描述
delay	Integer	聚焦前等待的毫秒数。
callback	Function()	元素被聚焦后要调用的函数。

该插件扩展自 jQuery 内置的 .focus() 方法。如果 jQuery UI 未加载，调用 `.focus()` 方法不会直接失败，因为该方法在 jQuery 中存在。但是不会发生预期的行为。

jQuery UI API - .position()

所属类别

方法重载（Method Overrides） | 方法（Methods） | 实用工具（Utilities）

用法

描述：相对另一个元素定位一个元素。

返回：jQuery

版本新增：1.8

```
.position( options )
```

参数	类型	描述
options	Object	

- **my**（默认值：`"center"`） 类型：String 描述：定义被定位元素上对准目标元素的位置："horizontal vertical" 对齐方式。一个单一的值，比如 "right" 将规范为 "right center"，"top" 将规范为 "center top"（下面的 CSS 公约）。可接受的水平值："left"、"center"、"right"。可接受的垂直值："top"、"center"、"bottom"。例如，"left top" 或 "center center"。每个纬度也可以包含偏移，以像素计或以百分比计，例如 "right+10 top-25%"。百分比偏移是相对于被定位的元素。
- **at**（默认值：`"center"`） 类型：String 描述：定义目标元素上对准被定位元素的位置："horizontal vertical" 对齐方式。如需了解更多细节请查看 my 选项上的可能值。百分比偏移是相对于目标元素。
- **of**（默认值：`null`） 类型：Selector 或 Element 或 jQuery 或 Event 描述：要定位的元素。如果您提供一个选择器（Selector）或 jQuery 对象，将使用第一个匹配元素。如果您提供一个事件（Event）对象，将使用 pageX 和 pageY 属性，例如 "#top-menu"。
- **collision**（默认值：`"flip"`） 类型：String 描述：当被定位元素在某些方向上溢出窗口，则移动它到另一个位置。与 my 和 at 选项相似，该选项会接受一个单一的值或一对 horizontal/vertical 值。例如："flip"、"fit"、"fit flip"、"fit none"。
  - "flip"：翻转元素到目标的相对一边，再次运行 collision 检测一遍查看元素是否适合。无论哪一边允许更多的元素可见，则使用那一边。
  - "fit"：把元素从窗口的边缘移开。
  - "flipfit"：首先应用 flip 逻辑，把元素放置在允许更多元素可见的那一边。然后应用 fit 逻辑，确保尽可能多的元素可见。

  - "none"：不应用任何 collision 检测。
- **using**（默认值：`null`） 类型：Function() 描述：当指定了该选项，实际属性设置则委托给该回调。接受两个参数：第一个是位置 top 和 left 值的哈希，可被转发到 .css() 或 .animate()；第二个提供了关于两个元素的位置和尺寸的反馈，同时也计算它们的相对位置。target 和 element 都有下列属性：element、left、top、width、height。另外，还有 horizontal、vertical 和 important，提供了十二个可能的方向，如 { horizontal: "center", vertical: "left", important: "horizontal" }。
- **within**（默认值：`window`） 类型：Selector 或 Element 或 jQuery 描述：元素定位为 within，会影响 collision 检测。如果您提供了一个选择器（Selector）或 jQuery 对象，则使用第一个匹配的元素。

jQuery UI `.position()` 方法允许您相对窗体（window）、文档、另一个元素或指针（cursor）/鼠标（mouse）来定位一个元素，无需考虑相对父元素的偏移（offset）。

注释：jQuery UI 不支持定位隐藏元素。

这是一个独立的 jQuery 插件，且对其他 jQuery UI 组件没有依赖关系。

该插件扩展自 jQuery 内置的 .position() 方法。如果 jQuery UI 未加载，调用 `.position()` 方法不会直接失败，因为该方法在 jQuery 中存在。但是不会发生预期的行为。

实例

一个简单的 jQuery UI 定位（Position）实例。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>.position() 实例</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  .positionDiv {
    position: absolute;
    width: 75px;
    height: 75px;
    background: green;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<div id="targetElement">
  <div class="positionDiv" id="position1"></div>
  <div class="positionDiv" id="position2"></div>
```

```
  <div class="positionDiv" id="position3"></div>
  <div class="positionDiv" id="position4"></div>
</div>

<script>
$( "#position1" ).position({
  my: "center",
  at: "center",
  of: "#targetElement"
});

$( "#position2" ).position({
  my: "left top",
  at: "left top",
  of: "#targetElement"
});

$( "#position3" ).position({
  my: "right center",
  at: "right bottom",
  of: "#targetElement"
});

$( document ).mousemove(function( event ) {
  $( "#position4" ).position({
    my: "left+3 bottom-3",
    of: event,
    collision: "fit"
  });
});
</script>

</body>
</html>
```

jQuery UI API 类别 - 方法（Methods）

尽管 jQuery UI 主要由 小部件（Widgets）、交互（Interactions） 和 特效（Effects） 组成，但是还有一些简单的方法，增加了便利性。

API	描述	也属于类别
.disableSelection()	禁用选择匹配的元素集合内的文本内容。	UI 核心（UI Core）
.effect()	对一个元素应用动画特效。	特效（Effects） | 特效核心（Effects Core）
.enableSelection()	启用选择匹配的元素集合内的文本内容。	UI 核心（UI Core）
.focus()	异步聚焦到一个元素。	方法重载（Method Overrides） | UI 核心（UI Core）
.hide()	使用自定义效果来隐藏匹配的元素。	特效（Effects） | 特效核心（Effects Core） | 方法重载（Method Overrides）
.position()	相对另一个元素定位一个元素。	方法重载（Method Overrides） | 实用工具（Utilities）
.removeUniqueId()	为匹配的元素集合移除由 .uniqueId() 设置的 Id。	UI 核心（UI Core）
.scrollParent()	获取最近的可滚动的祖先。	UI 核心（UI Core）
.show()	使用自定义效果来显示匹配的元素。	特效（Effects） | 特效核心（Effects Core） | 方法重载（Method Overrides）
.toggle()	使用自定义效果来显示或隐藏匹配的元素。	特效（Effects） | 特效核心（Effects Core） | 方法重载（Method Overrides）
.uniqueId()	为匹配的元素集合生成并申请一个唯一的 Id。	UI 核心（UI Core）
.zIndex()	为元素获取 z-index。	UI 核心（UI Core）

jQuery UI API - .disableSelection()

所属类别

方法（Methods） | UI 核心（UI Core）

用法

描述：禁用选择匹配的元素集合内的文本内容。

返回：jQuery

版本新增：1.6

版本废弃：1.9

```
.disableSelection()
```

该方法不接受任何参数。

禁用的文本选择是有害的，不建议使用。

jQuery UI API - .enableSelection()

所属类别

方法（Methods） | UI 核心（UI Core）

用法

描述：启用选择匹配的元素集合内的文本内容。

返回：jQuery

版本新增：1.6

版本废弃：1.9

```
.enableSelection()
```

该方法不接受任何参数。

`.enableSelection()` 方法可用于启用通过 `[.disableSelection()](api-disableSelection.html)` 禁用的文本选择。

jQuery UI API - .removeUniqueId()

所属类别

方法（Methods） | UI 核心（UI Core）

用法

描述：为匹配的元素集合移除由 .uniqueId() 设置的 Id。

返回：jQuery

版本新增：1.9

```
.removeUniqueId()
```

该方法不接受任何参数。

`.removeUniqueId()` 移除由 `[.uniqueId()](api-uniqueId.html)` 设置的 id。在未使用 `.uniqueId()` 设置 id 的元素上调用 `.removeUniqueId()` 则无影响，即使该元素有一个 id。

jQuery UI API - .scrollParent()

所属类别

方法（Methods） | UI 核心（UI Core）

用法

描述：获取最近的可滚动的祖先。

返回：jQuery

```
.scrollParent()
```

该方法不接受任何参数。

该方法查找最近的可滚动的祖先。换句话说， `.scrollParent()` 查找当前所选元素在其内滚动的元素。

该方法只在包含一个元素的 jQuery 对象上工作。

jQuery UI API - .uniqueId()

所属类别

方法（Methods） | UI 核心（UI Core）

用法

描述：为匹配的元素集合生成并申请一个唯一的 Id。

返回：jQuery

版本新增：1.9

```
.uniqueId()
```

该方法不接受任何参数。

许多小部件需要元素生成唯一的 id。 `.uniqueId()` 会检查元素是否有 id，如果元素没有 id，它将生成一个 id，并设置为该元素的 id。在未检查元素是否具有 id 就调用 `.uniqueId()` 是安全的。当小部件使用后需要清除，如果 id 是通过 `.uniqueId()` 添加的， `[.removeUniqueId()](api-removeUniqueId.html)` 方法将从元素上移除 id，如果 id 不是通过 `.uniqueId()` 添加的，则无影响。 `.removeUniqueId()` 之所以能区分 id，是因为 `.uniqueId()` 生成的 id 带有前缀 `"ui-id-"` 。

jQuery UI API - .zIndex()

所属类别

方法（Methods） | UI 核心（UI Core）

.zIndex() 用法

描述：为元素获取 z-index。

返回：jQuery

```
.zIndex()
```

该方法不接受任何参数。

`.zIndex()` 方法在查找一个元素的 z-index 时非常有用，忽略 z-index 是否是直接设置在元素上还是设置在祖先元素上。为了确定 z-index，该方法会在指定的元素上开始，且会沿着 DOM 查找，直到找到一个带有 z-index 的已定位的元素。如果未找到这样的元素，该方法将返回一个 `0` 值。

该方法假设带有嵌套 z-index 的元素不带有一个 `0` 值的 z-index。例如，给出下面的 DOM，内部元素将被当成不带有 z-index，因为在 Internet Explorer 中无法区分一个 `0` 显式值和无值。

```
<div style="z-index: -10;">
  <div style="z-index: 0;"></div>
</div>
```

.zIndex(zIndex) 用法

描述：为元素设置 z-index。

返回：Integer

```
.zIndex( zIndex )
```

参数	类型	描述
zIndex	Integer	要设置的 z-index。

这相当于 `.css( "zIndex", zIndex )` 。

jQuery UI API 类别 - 选择器（Selectors）

API	描述	也属于类别
:data() Selector	选择数据已存储在指定的键下的元素。	UI 核心（UI Core）
:focusable Selector	选择可被聚焦的元素。	UI 核心（UI Core）
:tabbable Selector	选择用户可通过 tab 键聚焦的元素。	UI 核心（UI Core）

jQuery UI API - :data() Selector

所属类别

选择器（Selectors） | UI 核心（UI Core）

用法

描述：选择数据已存储在指定的键下的元素。

```
jQuery( ":data(key)" )
```

参数	描述
key	数据的键。

表达式 `$( "div:data(foo)")` 匹配一个通过 `.data( "foo", value )` 存储数据的 `div` 。

实例

选择带有数据的元素，改变它们的背景颜色。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>:data() Selector 实例</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  div {
    width: 100px;
    height: 100px;
    border: 1px solid #000;
    float: left;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<div id="one"></div>
<div id="two"></div>
<div id="three"></div>

<script>
$( "#one" ).data( "color", "blue" );
$( "#three" ).data( "color", "green" );

$( ":data(color)" ).each(function() {
  var element = $( this );
  element.css( "backgroundColor", element.data( "color" ) );
});
</script>

</body>
</html>
```

jQuery UI API - :focusable Selector

所属类别

选择器（Selectors） | UI 核心（UI Core）

用法

描述：选择可被聚焦的元素。

```
jQuery( ":focusable" )
```

一些元素本身是可聚焦的（focusable），而另一些元素需要显式设置 tab 索引。以上两种情况，为了可聚焦（focusable），元素都必须是可见的。

下面类型的元素如果未被禁用，则是可聚焦的（focusable）： `input` 、 `select` 、 `textarea` 、 `button` 和 `object` 。锚如果带有 `href` 或 `tabindex` 属性，则是可聚焦的（focusable）。 `area` 元素如果在一个已命名的 map 内，且带有 `href` 属性，且有一个可见的图像使用了该 map，则是可聚焦的（focusable）。所有其他完全基于 `tabindex` 属性和可见度的元素是可聚焦的（focusable）。

注释：带有负的 tab 索引的元素是 `:focusable` ，不是 `[:tabbable](api-tabbable-selector.html)` 。

实例

选择可聚焦的元素，且用一个红色边框突出显示。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>:focusable Selector 实例</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  input, a, p {
    border: 1px solid #000;
  }
  div {
    padding: 5px;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<div><input value="文本输入"></div>
<div><a>不带有 href 的锚</a></div>
<div><a href="#">带有 href 的锚</a></div>
<div><p>不带有 tabindex 的段落</p></div>
<div><p tabindex="1">带有 tabindex 的段落</p></div>

<script>
$( ":focusable" ).css( "border-color", "red" );
</script>

</body>
</html>
```

jQuery UI API - :tabbable Selector

所属类别

选择器（Selectors） | UI 核心（UI Core）

用法

描述：选择用户可通过 tab 键聚焦的元素。

```
jQuery( ":tabbable" )
```

一些元素本身是可通过 tab 键聚焦的（tabbable），而另一些元素需要显式设置一个正的 tab 索引。以上两种情况，为了可通过 tab 键聚焦（tabbable），元素都必须是可见的。

下面类型的元素如果不带有负的 tab 索引且未被禁用，则是可通过 tab 键聚焦的（tabbable）：`input`、`select`、`textarea`、`button` 和 `object`。锚如果带有 `href` 或正的 `tabindex` 属性，则是可通过 tab 键聚焦的（tabbable）。`area` 元素如果在一个已命名的 map 内，且带有 `href` 属性，且有一个可见的图像使用了该 map，则是可通过 tab 键聚焦的（tabbable）。所有其他完全基于 `tabindex` 属性和可见度的元素是可通过 tab 键聚焦的（tabbable）。

注释：带有负的 tab 索引的元素是 `[:focusable](api-focusable-selector.html)`，不是 `:tabbable`。

实例

选择可通过 tab 键聚焦的元素，且用一个红色边框突出显示。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>:tabbable Selector 实例</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  input {
    border: 1px solid #000;
  }
  div {
    padding: 5px;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<div><input value="没有 tabindex"></div>
<div><input tabindex="5" value="正的 tabindex"></div>
<div><input tabindex="-1" value="负的 tabindex"></div>

<script>
$( ":tabbable" ).css( "border-color", "red" );
</script>

</body>
</html>
```

jQuery UI API 类别 - 主题（Theming）

jQuery UI 包括一个强大的 CSS 框架，用于创建自定义的 jQuery 小部件。该框架包括涵盖广泛的公共用户界面需求的 Class，并且可以使用 jQuery UI ThemeRoller 进行操纵。通过使用 jQuery UI CSS 框架创建您自己的 UI 组件，您应采用共享标记公约，便于插件社区的代码集成。您可以查看 jQuery UI 主题 了解更多详情。

API	描述
CSS 框架（CSS Framework）	jQuery UI 使用的允许组件主题化的 Class 名称列表。
图标（Icons）	jQuery UI 提供的图标列表。
堆叠元素（Stacking Elements）	一种处理 z-index 和堆叠元素的模式。

jQuery UI API - CSS 框架（CSS Framework）

下面是 jQuery UI 使用的 Class 名称列表。这些 Class 用来创建跨应用程序的视觉一致性，且允许组件通过 jQuery UI ThemeRoller 进行主题化。下面的 CSS 类根据样式是否是固定的结构化的，或者是否是可主题化的（颜色、字体、背景等），分别定义在 ui.core.css 和 ui.theme.css 两个文件中。

布局助手

- `.ui-helper-hidden` ：对元素应用 `display: none` 。
- `.ui-helper-hidden-accessible` ：对元素应用访问隐藏（通过页面绝对定位）。
- `.ui-helper-reset` ：UI 元素的基本样式重置。重置的元素比如： `padding` 、 `margin` 、 `text-decoration` 、list-style，等等。
- `.ui-helper-clearfix` ：对父元素应用浮动包装属性。
- `.ui-helper-zfix` ：对 `<iframe>` 元素应用 iframe "fix" CSS。
- `.ui-front` ：应用 z-index 来管理屏幕上多个小部件的堆叠，如需了解更多详情，请查看 堆叠元素（Stacking Elements） 页面。

小部件容器

- `.ui-widget` ：对所有小部件的外部容器应用的 Class。对小部件应用字体和字体尺寸，同时也对自表单元素应用相同的字体和 1em 的字体尺寸，以应对 Windows 浏览器中的继承问题。
- `.ui-widget-header` ：对标题容器应用的 Class。对元素及其子元素的文本、链接、图标应用标题容器样式。
- `.ui-widget-content` ：对内容容器应用的 Class。对元素及其子元素的文本、链接、图标应用内容容器样式。（可应用到标题的父元素或者同级元素）

交互状态

- `.ui-state-default` ：对可点击按钮元素应用的 Class。对元素及其子元素的文本、链接、图标应用 "clickable default" 容器样式。
- `.ui-state-hover` ：当鼠标悬浮在可点击按钮元素上时应用的 Class。对元素及其子元素的文本、链接、图标应用 "clickable hover" 容器样式。
- `.ui-state-focus` ：当键盘聚焦在可点击按钮元素上时应用的 Class。对元素及其子元素的文本、链接、图标应用 "clickable hover" 容器样式。
- `.ui-state-active` ：当鼠标点击可点击按钮元素上时应用的 Class。对元素及其子元素的文本、链接、图标应用 "clickable active" 容器样式。

交互提示 Cues

- `.ui-state-highlight` ：对高亮或者选中元素应用的 Class。对元素及其子

元素的文本、链接、图标应用 "highlight" 容器样式。
- `.ui-state-error` ：对错误消息容器元素应用的 Class。对元素及其子元素的文本、链接、图标应用 "error" 容器样式。
- `.ui-state-error-text` ：对只有无背景的错误文本颜色应用的 Class。可用于表单标签，也可以对子图标应用错误图标颜色。
- `.ui-state-disabled` ：对禁用的 UI 元素应用一个暗淡的不透明度。意味着对一个已经定义样式的元素添加额外的样式。
- `.ui-priority-primary` ：对第一优先权的按钮应用的 Class。应用粗体文本。
- `.ui-priority-secondary` ：对第二优先权的按钮应用的 Class。应用正常粗细的文本，对元素应用轻微的透明度。

图标

状态和图像

- `.ui-icon` ：对图标元素应用的基本 Class。设置尺寸为 16px 方块，隐藏内部文本，对 "content" 状态的精灵图像设置背景图像。注意： *`.ui-icon` class* 将根据它的父容器得到一个不同的精灵背景图像。例如， *`ui-state-default`* 容器内的 *`ui-icon`* 元素将根据 *`ui-state-default`* 的图标颜色进行着色。

图标类型

在声明 `.ui-icon` class 之后，接着您可以声明一个秒速图标类型的 class。通常情况下，图标 class 遵循语法 `.ui-icon-{icon type}-{icon sub description}-{direction}` 。

例如，一个指向右侧的三角形图标，如下所示： `.ui-icon-triangle-1-e`

jQuery UI 的 ThemeRoller 在它的预览一栏中提供了全套的 CSS 框架图标。将鼠标悬浮在图标上可查看 class 名称。

其他视觉效果

圆角半径助手

- `.ui-corner-tl` ：对元素的左上角应用圆角半径。
- `.ui-corner-tr` ：对元素的右上角应用圆角半径。
- `.ui-corner-bl` ：对元素的左下角应用圆角半径。
- `.ui-corner-br` ：对元素的右下角应用圆角半径。
- `.ui-corner-top` ：对元素上边的左右角应用圆角半径。
- `.ui-corner-bottom` ：对元素下边的左右角应用圆角半径。
- `.ui-corner-right` ：对元素右边的上下角应用圆角半径。
- `.ui-corner-left` ：对元素左边的上下角应用圆角半径。

- `.ui-corner-all`：对元素的所有四个角应用圆角半径。

覆盖 & 阴影

- `.ui-widget-overlay`：对覆盖屏幕应用 100% 宽度和高度，同时设置背景颜色/纹理和屏幕不透明度。
- `.ui-widget-shadow`：对覆盖应用的 Class，设置了不透明度、上偏移/左偏移，以及阴影的 "厚度"。厚度是通过对阴影所有边设置内边距（padding）进行应用的。偏移是通过设置上外边距（margin）和左外边距（margin）进行应用的（可以是正数，也可以是负数）。

jQuery UI API - 图标（Icons）

jQuery UI 提供了大量可以通过对元素应用 Class 名称来使用的图标（Icons）。Class 名称大体上遵循语法 `.ui-icon-{icon type}-{icon sub description}-{direction}`。例如，下面将显示一个向北的厚箭头的图标：

```
<span class="ui-icon ui-icon-arrowthick-1-n"></span>
```

图标也集成到一些 jQuery UI 的小部件，比如 accordion、button、menu。

下面是 jQuery UI 提供的图标的完整列表：

jQuery UI API - 堆叠元素（Stacking Elements）

堆叠的或者移动到其他元素前面的小部件（Widgets）当放置到现实世界的页面中时经常面临挑战。通常通过简单地改变堆叠元素的 `z-index` 或者父元素来避免页面上的冲突。但是，jQuery UI 需要一个不需要手动改变 `z-index` 值的通用的解决方案。这是通过 `ui-front` class 来完成的，通常还伴随着堆叠组件上的 `appendTo` 选项。

`ui-front` class

`ui-front` class 是非常基础的。它只是在元素上设置了一个静态的 `z-index` 值。但是，class 的存在是用来表明堆叠元素要追加到哪里。这允许我们利用嵌套层内容，生成一个在大多数情况下都能使用的默认的 DOM 位置。

注释：当使用 *`ui-front`* 时，您必须设置 *`position`* 为 *`relative`*、*`absolute`* 或 *`fixed`*，以便应用 *`z-index`*。

堆叠技术（stacking technique）

追加堆叠元素到页面的任何小部件都必须使用 `ui-front` class，且在大多数情况下，应该有一个 `appendTo` 选项。堆叠元素应遵循下面的规则：

- 如果一个值设置为 `appendTo` 选项，则追加堆叠元素到指定的元素。
- 如果 `appendTo` 选项被设置为 `null`（默认），则小部件应从相关的元素开始遍历 DOM。例如，当自动完成（autocomplete）菜单被追加到 DOM，遍历则从相关的 input 元素开始。
  - 如果找到一个带有 `ui-front` class 的元素，则追加到该元素。
  - 如果没有找到一个带有 `ui-front` class 的元素，则追加到主体（body）。
- 堆叠元素的 `position` 必须设置为 `relative`、`absolute` 或 `fixed`，以便应用来自 `ui-front` class 的 `z-index`。使用 .position() 将自动设置 `position`。

jQuery UI API 类别 - UI 核心（UI Core）

由 jquery.ui.core.js 提供的功能。

API	描述	也属于类别
:data() Selector	选择数据已存储在指定的键下的元素。	选择器（Selectors）
.disableSelection()	禁用选择匹配的元素集合内的文本内容。	方法（Methods）
.enableSelection()	启用选择匹配的元素集合内的文本内容。	方法（Methods）
.focus()	异步聚焦到一个元素。	方法重载（Method Overrides） | 方法（Methods）
:focusable Selector	选择可被聚焦的元素。	选择器（Selectors）
jQuery.ui.keyCode	一个相对于数字值的关键代码描述的映射。	
.removeUniqueId()	为匹配的元素集合移除由 .uniqueId() 设置的 Id。	方法（Methods）
.scrollParent()	获取最近的可滚动的祖先。	方法（Methods）
:tabbable Selector	选择用户可通过 tab 键聚焦的元素。	选择器（Selectors）
.uniqueId()	为匹配的元素集合生成并申请一个唯一的 Id。	方法（Methods）
.zIndex()	为元素获取 z-index。	方法（Methods）

jQuery UI API 类别 - 实用工具（Utilities）

API	描述	也属于类别
Easings	Easing 函数指定动画在不同点上的行进速度。	
部件库（Widget Factory）	使用与所有 jQuery UI 小部件相同的抽象化来创建有状态的 jQuery 插件。	小部件（Widgets）
插件桥（Widget Plugin Bridge）	jQuery.widget.bridge() 方法是 jQuery 部件库（Widget Factory）的一部分。它扮演着由 $.widget() 创建的对象和 jQuery API 之间的中介。	小部件（Widgets）
鼠标交互（Mouse Interaction）	基本交互层。	交互（Interactions）
.position()	相对另一个元素定位一个元素。	方法重载（Method Overrides） | 方法（Method）

jQuery UI API - 部件库（Widget Factory）

所属类别

实用工具（Utilities） | 小部件（Widgets）

jQuery.widget(name [, base], prototype) 用法

描述：使用与所有 jQuery UI 小部件相同的抽象化来创建有状态的 jQuery 插件。

```
jQuery.widget( name [, base ], prototype )
```

参数	类型	类型
name	String	要创建的小部件名称，包括命名空间。
base	Function()	要继承的基础小部件。必须是一个可以使用 `new` 关键词实例化的构造函数。默认为 jQuery.Widget。
prototype	PlainObject	要作为小部件原型使用的对象。

您可以使用 `$.Widget` 对象作为要继承的基础，或者可以明确地从现有的 jQuery UI 或第三方控件，从头开始创建新的小部件。定义一个带有相同名称的小部件来继承基础部件，甚至允许您适当地扩展小部件。

jQuery UI 中包含许多保持状态的小部件，因此比典型的 jQuery 插件稍有不同的使用模式。所有的jQuery UI 小部件使用相同的模式，这是由部件库（Widget Factory）定义的。所以，只要您学会使用其中一个，您就知道如何使用其他的小部件（Widget）。

注释：本章节使用 进度条部件（Progressbar Widget） 演示实例，但是语法适用于每个小部件。

初始化

为了跟踪小部件的状态，我们必须引入小部件的全生命周期。小部件初始化时生命周期开始。要初始化一个小部件，我们只需要简单地在一个或多个元素上调用插件。

```
$( "#elem" ).progressbar();
```

这将初始化 jQuery 对象中的每个元素。上面实例中元素 id 为 `"elem"` 。

选项

由于 `progressbar()` 调用时不带参数，小部件是使用默认选项进行初始化的。我们可以在初始化时传递一组选项来覆盖默认选项：

```
$( "#elem" ).progressbar({ value: 20 });
```

我们可以根据需要传递选项的个数，任何我们未传递的选项都使用它们的默认值。

您可以传递多个选项参数，这些参数将会被合并为一个对象（类似于 `$.extend( true, target, object1, objectN )`）。这在为所有实例覆盖一些设置，实例间共享选项时很有用：

```
var options = { modal: true, show: "slow" };
$( "#dialog1" ).dialog( options );
$( "#dialog2" ).dialog( options, { autoOpen: false });
```

所有在初始化时传递的选项都是深拷贝的，确保后续在不影响小部件的情况下修改对象。数组是唯一的例外，它们是按原样引用的。这个例外是为了适当地支持数据绑定，其中数据源必须作为引用。

默认值保存在小部件的属性中，因此我们可以覆盖 jQuery UI 设置的值。例如，在下面的设置后，所有将来的进度条实例将默认为值 80：

```
$.ui.progressbar.prototype.options.value = 80;
```

选项是小部件状态的组成部分，所以我们也可以在初始化后设置选项。我们会在后续看到 option 方法。

方法

现在小部件已经初始化，我们可以查询它的状态，或者在小部件上执行动作。所有初始化后的动作都是以方法调用方式执行。为了在小部件上调用一个方法，我们向 jQuery 插件传递方法的名称。例如，在进度条部件（Progressbar Widget）上调用 `value()` 方法，我们可以使用：

```
$( "#elem" ).progressbar( "value" );
```

如果方法接受参数，我们可以在方法名称后传递参数。例如，要传递参数 `40` 到 `value()` 方法，我们可以使用：

```
$( "#elem" ).progressbar( "value", 40 );
```

就像 jQuery 中的其他方法，大多数的小部件方法返回 jQuery 对象：

```
$( "#elem" )
  .progressbar( "value", 90 )
  .addClass( "almost-done" );
```

每个小部件都有自己的方法设置，这些设置是基于小部件提供的功能。但是，有一些方法是存在于所有的小部件上，这会在下面进行详细讲解。

事件

所有的小部件都有与它们各种行为相关的事件，以便在状态改变的时候通知您。对于大多数的小部件，当事件被触发时，名称以小部件名称的小写字母形式作为前缀。例如，我们可以绑定进度条的 `change` 事件，该事件在值改变时触发。

```
$( "#elem" ).bind( "progressbarchange", function() {
  alert( "The value has changed!" );
});
```

每个事件都有一个对应的回调，这会作为选项。如果需要，我们可以抓住进度条的 `change` 回调，而不用绑定 `progressbarchange` 事件。

```
$( "#elem" ).progressbar({
  change: function() {
    alert( "The value has changed!" );
  }
});
```

所有的小部件都有一个 `change` 事件，该事件在实例化时触发。

实例化

小部件的实例是使用带有小部件全称作为键的 `jQuery.data()` 存储的。因此，您可以使用下面代码从元素检索进度条部件（Progressbar Widget）的实例对象。

```
$( "#elem" ).data( "ui-progressbar" );
```

元素是否绑定了给定小部件，可以使用 `:data` 选择器来检测。

```
$( "#elem" ).is( ":data( 'ui-progressbar' )" ); // true
$( "#elem" ).is( ":data( 'ui-draggable' )" ); // false
```

您也可以使用 `:data` 来获得作为给定小部件实例的所有元素的列表。

```
$( ":data( 'ui-progressbar' )" );
```

属性

所有的小部件都有下面的属性：

- **defaultElement**：当构造小部件实例未提供元素时要使用的元素。例如，由于进度条的 `defaultElement` 是 `"<div> "`， `$.ui.progressbar({ value: 50 })` 在一个新建的 `<div>` 上实例化进度条小部件实例。
- **document**：其内包含小部件元素的 `document` 。如果需要在框架内与小部件进行交互时很有用。
- **element**：一个 jQuery 对象，包含用于实例化小部件的 元素。如果您选择多个元素并调用 `.myWidget()` ，将为每个元素创建一个单独的小部件实例。因此，该属性总是包含一个元素。
- **namespace**：小部件原型存储的全局 jQuery 对象的位置。例如， `"ui"` 的 `namespace` 表示小部件原型存储在 `$.ui` 。
- **options**：一个包含小部件当前使用选项的对象。在实例化时，用户提供的任何选项将会自动与 `$.myNamespace.myWidget.prototype.options` 中定义的默认值合并。用户指定的选项会覆盖默认值。
- **uuid**：一个表示控件标识符的唯一整数。
- **version**：小部件的字符串版本。对于 jQuery UI 小部件，该属性会被设置为小部件使用的 jQuery UI 的版本。插件开发者必须在他们的原型中明确设置该属性。
- **widgetEventPrefix**：添加到小部件事件名称的前缀。例如，可拖拽小部件（Draggable Widget）的 `widgetEventPrefix` 是 `"drag"` ，因此当创建一个 draggable 时，事件的名称是 `"dragcreate"` 。默认情况下，小部件的 `widgetEventPrefix` 是它的名称。注意：该属性已被废弃，将在以后的版本中非常。事件名称将被改为 *widgetName:eventName* （例如 *`"draggable:create"`* ）。
- **widgetFullName**：包含命名空间的小部件全称。对于 `$.widget( "myNamespace.myWidget", {} )` ， `widgetFullName` 将是 `"myNamespace-myWidget"` 。
- **widgetName**：小部件的名称。对于 `$.widget( "myNamespace.myWidget", {} )` ， `widgetName` 将是 `"myWidget"` 。
- **window**：其内包含小部件元素的 `window` 。如果需要在框架内与小部件进行交互时很有用。

jQuery.Widget 基础小部件用法

描述：部件库（Widget Factory）使用的基础小部件。

快速导航

选项	
disabledhideshow	_create_delay_destroy_focusable_getCreateEventData_ge

disabled

类型：Boolean

描述：如果设置为 `true` ，则禁用该小部件。

代码实例：

初始化带有指定 `disabled` 选项的小部件：

```
$( ".selector" ).widget({ disabled: true });
```

在初始化后，获取或设置 `disabled` 选项：

```
// getter
var disabled = $( ".selector" ).widget( "option", "disabled" );

// setter
$( ".selector" ).widget( "option", "disabled", true );
```

默认值：false

hide

类型：Boolean 或 Number 或 String 或 Object

描述：是否使用动画隐藏元素，以及如何动画隐藏元素。

支持多个类型：

- **Boolean**：当设置为 `false` 时，则不使用动画，元素会立即隐藏。当设置为 `true` 时，元素会使用默认的持续时间和默认的 easing 淡出。
- **Number**：元素将使用指定的持续时间和默认的 easing 淡出。
- **String**：元素将使用指定的特效隐藏。该值可以是一个内建的 jQuery 动画方法名称，比如 `"slideUp"` ，也可以是一个 jQuery UI 特效 的名称，比如 `"fold"` 。以上两种情况的特效将使用默认的持续时间和默认的 easing。
- **Object**：如果值是一个对象，则 `effect` 、 `delay` 、 `duration` 和 `easing` 属性会被提供。如果 `effect` 属性包含 jQuery 方法的名称，则使用该方法，否则，它被认为是一个 jQuery UI 特效的名称。当使用一个支持额外设置的 jQuery UI 特效时，您可以在对象中包含这些设置，且它们会被传递

给特效。如果 `duration` 或 `easing` 被省略，则使用默认值。如果 `effect` 被省略，则使用 `"fadeOut"` 。如果 `delay` 被省略，则不使用延迟。

代码实例：

初始化带有指定 `hide` 选项的小部件：

```
$( ".selector" ).widget({ hide: { effect: "explode", duration: 100(
```

在初始化后，获取或设置 `hide` 选项：

```
// getter
var hide = $( ".selector" ).widget( "option", "hide" );

// setter
$( ".selector" ).widget( "option", "hide", { effect: "explode", du
```

默认值：null

show

类型：Boolean 或 Number 或 String 或 Object

描述：是否使用动画显示元素，以及如何动画显示元素。

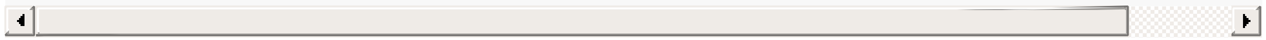
支持多个类型：

- **Boolean**：当设置为 `false` 时，则不使用动画，元素会立即显示。当设置为 `true` 时，元素会使用默认的持续时间和默认的 easing 淡入。
- **Number**：元素将使用指定的持续时间和默认的 easing 淡入。
- **String**：元素将使用指定的特效显示。该值可以是一个内建的 jQuery 动画方法名称，比如 `"slideDown"` ，也可以是一个 jQuery UI 特效 的名称，比如 `"fold"` 。以上两种情况的特效将使用默认的持续时间和默认的 easing。
- **Object**：如果值是一个对象，则 `effect` 、 `delay` 、 `duration` 和 `easing` 属性会被提供。如果 `effect` 属性包含 jQuery 方法的名称，则使用该方法，否则，它被认为是一个 jQuery UI 特效的名称。当使用一个支持额外设置的 jQuery UI 特效时，您可以在对象中包含这些设置，且它们会被传递给特效。如果 `duration` 或 `easing` 被省略，则使用默认值。如果 `effect` 被省略，则使用 `"fadeIn"` 。如果 `delay` 被省略，则不使用延迟。

代码实例：

初始化带有指定 `show` 选项的小部件：

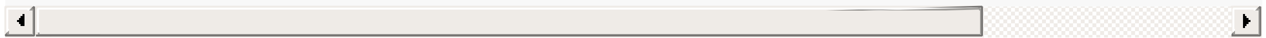
```
$( ".selector" ).widget({ show: { effect: "blind", duration: 800 }
```



在初始化后，获取或设置 `show` 选项：

```
// getter
var show = $( ".selector" ).widget( "option", "show" );

// setter
$( ".selector" ).widget( "option", "show", { effect: "blind", durat
```



默认值：null

方法

_create()

类型：jQuery (plugin only)

描述：`_create()` 方法是小部件的构造函数。没有参数，但是 `this.element` 和 `this.options` 已经设置。

- 该方法不接受任何参数。

代码实例：

基于一个选项设置小部件元素的背景颜色。

```
_create: function() {
  this.element.css( "background-color", this.options.color );
}
```

_delay(fn [, delay])

类型：Number

描述：在指定延迟后调用提供的函数。保持 `this` 上下文正确。本质上是 `setTimeout()`。

使用 `clearTimeout()` 返回超时 ID。

- **fn** 类型：Function() 或 String 描述：要调用的函数。也可以是小部件上方法的名称。
- **delay** 类型：Number 描述：调用函数前等待的毫秒数，默认为 `0`。

代码实例：

100 毫秒后在小部件上调用 `_foo()` 方法。

```
this._delay( this._foo, 100 );
```

_destroy()

类型：jQuery (plugin only)

描述：公共的 `destroy()` 方法清除所有的公共数据、事件等等。代表了定制、指定小部件、清理的 `_destroy()` 。

- 该方法不接受任何参数。

代码实例：

当小部件被销毁时，从小部件的元素移除一个 class。

```
_destroy: function() {
  this.element.removeClass( "my-widget" );
}
```

_focusable(element)

类型：jQuery (plugin only)

描述：建立聚焦在元素上时要应用 `ui-state-focus` class 的 `element` 。

- **element** 类型：jQuery 描述：要应用 focusable 行为的元素。

代码实例：

向小部件内的一组元素应用 focusable 样式：

```
this._focusable( this.element.find( ".my-items" ) );
```

_getCreateEventData()

类型：Object

描述：所有的小部件触发 `create` 事件。默认情况下，事件中不提供任何的数据，但是该方法会返回一个对象，作为 `create` 事件的数据被传递。

- 该方法不接受任何参数。

代码实例：

向 `create` 事件处理程序传递小部件的选项，作为参数。

```
_getCreateEventData: function() {
  return this.options;
}
```

_getCreateOptions()

类型：Object

描述：该方法允许小部件在初始化期间为定义选项定义一个自定义的方法。用户提供的选项会覆盖该方法返回的选项，即会覆盖默认的选项。

- 该方法不接受任何参数。

代码实例：

让小部件元素的 id 属性作为选项可用。

```
_getCreateOptions: function() {
  return { id: this.element.attr( "id" ) };
}
```

_hide(element, option [, callback])

类型：jQuery (plugin only)

描述：使用内置的动画方法或使用自定义的特效隐藏一个元素。如需了解可能的 `option` 值，请查看 hide。

- **element** 类型：jQuery 描述：要隐藏的元素。
- **option** 类型：Object 描述：定义如何隐藏元素的设置。
- **callback** 类型：Function() 描述：元素完全隐藏后要调用的回调。

代码实例：

为自定义动画传递 `hide` 选项。

```
this._hide( this.element, this.options.hide, function() {

  // Remove the element from the DOM when it's fully hidden.
  $( this ).remove();
});
```

_hoverable(element)

类型：jQuery (plugin only)

描述：建立悬浮在元素上时要应用 `ui-state-hover` class 的 `element` 。事件处理程序在销毁时自动清理。

- **element** 类型：jQuery 描述：要应用 hoverable 行为的元素。

代码实例：

当悬浮在元素上时，向元素内所有的 `div` 应用 hoverable 样式。

```
this._hoverable( this.element.find( "div" ) );
```

_init()

类型：jQuery (plugin only)

描述：小部件初始化的理念与创建不同。任何时候不带参数的调用插件或者只带一个选项哈希的调用插件，初始化小部件。当小部件被创建时会包含这个方法。

注释：如果存在不带参数成功调用小部件时要执行的逻辑动作，初始化只能在这时处理。

- 该方法不接受任何参数。

代码实例：

如果设置了 `autoOpen` 选项，则调用 `open()` 方法。

```
_init: function() {
  if ( this.options.autoOpen ) {
    this.open();
  }
}
```

_off(element, eventName)

类型：jQuery (plugin only)

描述：从指定的元素取消绑定事件处理程序。

- **element** 类型：jQuery 描述：要取消绑定事件处理程序的元素。不像 `_on()` 方法， `_off()` 方法中元素是必需的。
- **eventName** 类型：String 描述：一个或多个空格分隔的事件类型。

代码实例：

从小部件的元素上取消绑定所有 click 事件。

```
this._off( this.element, "click" );
```

_on([suppressDisabledCheck] [, element], handlers)

类型：jQuery (plugin only)

描述：授权通过事件名称内的选择器被支持，例如 `"click .foo"` 。 `_on()` 方法提供了一些直接事件绑定的好处：

- 保持处理程序内适当的 `this` 上下文。
- 自动处理禁用的部件：如果小部件被禁用或事件发生在带有 `ui-state-disabled` class 的元素上，则不调用事件处理程序。可以被 `suppressDisabledCheck` 参数重写。
- 事件处理程序会自动添加命名空间，在销毁时会自动清理。
- **suppressDisabledCheck**（默认值： `false` ） 类型：Boolean 描述：是否要绕过禁用的检查。
- **element** 类型：jQuery 描述：要绑定事件处理程序的元素。如果未提供元素， `this.element` 用于未授权的事件，widget 元素 用于授权的事件。
- **handlers** 类型：Object 描述：一个 map，其中字符串键代表事件类型，可选的选择器用于授权，值代表事件调用的处理函数。

代码实例：

放置小部件元素内所有被点击的链接的默认行为。

```
this._on( this.element, {
  "click a": function( event ) {
    event.preventDefault();
  }
});
```

_setOption(key, value)

类型：jQuery (plugin only)

描述：为每个独立的选项调用 `_setOptions()` 方法。小部件状态随着改变而更新。

- **key** 类型：String 描述：要设置的选项名称。
- **value** 类型：Object 描述：为选项设置的值。

代码实例：

当小部件的 `height` 或 `width` 选项改变时，更新小部件的元素。

```
_setOption: function( key, value ) {
  if ( key === "width" ) {
    this.element.width( value );
  }
  if ( key === "height" ) {
    this.element.height( value );
  }
  this._super( key, value );
}
```

_setOptions(options)

类型：jQuery (plugin only)

描述：当调用 `option()` 方法时调用，无论以什么形式调用 `option()` 。如果您要根据多个选项的改变而改变处理器密集型，重载该方法是很有用的。

- **options** 类型：Object 描述：为选项设置的值。

代码实例：

如果小部件的 `height` 或 `width` 选项改变，调用 `resize` 方法。

```
_setOptions: function( options ) {
  var that = this,
    resize = false;

  $.each( options, function( key, value ) {
    that._setOption( key, value );
    if ( key === "height" &#124;&#124; key === "width" ) {
      resize = true;
    }
  });

  if ( resize ) {
    this.resize();
  }
}
```

_show(element, option [, callback])

类型：jQuery (plugin only)

描述：使用内置的动画方法或使用自定义的特效显示一个元素。如需了解可能的 `option` 值，请查看 show。

- **element** 类型：jQuery 描述：要显示的元素。
- **option** 类型：Object 描述：定义如何显示元素的设置。

- **callback** 类型：Function() 描述：元素完全显示后要调用的回调。

代码实例：

为自定义动画传递 `show` 选项。

```
this._show( this.element, this.options.show, function() {

  // Focus the element when it's fully visible.
  this.focus();
}
```

_super([arg] [, ...])

类型：jQuery (plugin only)

描述：从父部件中调用相同名称的方法，带有任意指定的参数。本质上是 `.call()` 。

- **arg** 类型：Object 描述：要传递给父部件的方法的零到多个参数。

代码实例：

处理 `title` 选项更新，并调用付部件的 `_setOption()` 来更新选项的内部存储。

```
_setOption: function( key, value ) {
  if ( key === "title" ) {
    this.element.find( "h3" ).text( value );
  }
  this._super( key, value );
}
```

_superApply(arguments)

类型：jQuery (plugin only)

描述：从父部件中调用相同名称的方法，带有参数的数组。本质上是 `.apply()` 。

- **arguments** 类型：Array 描述：要传递给父部件的方法的参数数组。

代码实例：

处理 `title` 选项更新，并调用付部件的 `_setOption()` 来更新选项的内部存储。

```
_setOption: function( key, value ) {
  if ( key === "title" ) {
    this.element.find( "h3" ).text( value );
  }
  this._superApply( arguments );
}
```

_trigger(type [, event] [, data])

类型：Boolean

描述：触发一个事件及其相关的回调。带有该名称的选项与作为回调被调用的类型相等。

事件名称是小部件名称和类型的小写字母串。

注释：当提供数据时，您必须提供所有三个参数。如果没有传递事件，则传递 `null` 。

如果默认行为是阻止的，则返回 `false` ，否则返回 `true` 。当处理程序返回 `false` 时或调用 `event.preventDefault()` 时，则阻止默认行为发生。

- **type** 类型：String 描述： `type` 应该匹配回调选项的名称。完整的事件类型会自动生成。
- **event** 类型：Event 描述：导致该事件发生的原始事件，想听众提供上下文时很有用。
- **data** 类型：Object 描述：一个与事件相关的数据哈希。

代码实例：

当按下一个键时，触发 `search` 事件。

```
this._on( this.element, {
  keydown: function( event ) {

    // Pass the original event so that the custom search event has
    // useful information, such as keyCode
    this._trigger( "search", event, {

      // Pass additional information unique to this event
      value: this.element.val()
    });
  }
});
```

destroy()

类型：jQuery (plugin only)

描述：完全移除小部件功能。这会把元素返回到它的预初始化状态。

- 该方法不接受任何参数。

代码实例：

当点击小部件的任意锚点时销毁小部件。

```
this._on( this.element, {
  "click a": function( event ) {
    event.preventDefault();
    this.destroy();
  }
});
```

disable()

类型：jQuery (plugin only)

描述：禁用小部件。

- 该方法不接受任何参数。

代码实例：

当点击小部件的任意锚点时禁用小部件。

```
this._on( this.element, {
  "click a": function( event ) {
    event.preventDefault();
    this.disable();
  }
});
```

enable()

类型：jQuery (plugin only)

描述：启用小部件。

- 该方法不接受任何参数。

代码实例：

当点击小部件的任意锚点时启用小部件。

```
this._on( this.element, {
  "click a": function( event ) {
    event.preventDefault();
    this.enable();
  }
});
```

option(optionName)

类型：Object

描述：获取当前与指定的 `optionName` 关联的值。

- **optionName** 类型：String 描述：要获取的选项的名称。

代码实例：

获得 `width` 选项的值。

```
this.option( "width" );
```

option()

类型：PlainObject

描述：获取一个包含键/值对的对象，键/值对表示当前小部件选项哈希。

- 该方法不接受任何参数。

代码实例：

记录每个小部件选项的键/值对，用于调试。

```
var options = this.option();
for ( var key in options ) {
  console.log( key, options[ key ] );
}
```

option(optionName, value)

类型：jQuery (plugin only)

描述：设置与指定的 `optionName` 关联的小部件选项的值。

- **optionName** 类型：String 描述：要设置的选项的名称。
- **value** 类型：Object 描述：要为选项设置的值。

代码实例：

设置 `width` 选项为 `500` 。

```
this.option( "width", 500 );
```

option(options)

类型：jQuery (plugin only)

描述：为小部件设置一个或多个选项。

- **options** 类型：Object 描述：要设置的 option-value 对。

代码实例：

设置 `height` 和 `width` 选项为 `500` 。

```
this.option({
  width: 500,
  height: 500
});
```

widget()

类型：jQuery

描述：返回一个包含原始元素或其他相关的生成元素的 `jQuery` 对象。

- 该方法不接受任何参数。

代码实例：

当创建小部件时，在小部件的原始元素周围放置一个红色的边框。

```
_create: function() {
  this.widget().css( "border", "2px solid red" );
}
```

事件

create(event, ui)

类型：widgetcreate

描述：当小部件被创建时触发。

- **event** 类型：Event
- **ui** 类型：Object

注意：`ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

代码实例：

初始化带有指定 create 回调的小部件：

```
$( ".selector" ).widget({
  create: function( event, ui ) {}
});
```

绑定一个事件监听器到 widgetcreate 事件：

```
$( ".selector" ).on( "widgetcreate", function( event, ui ) {} );
```

jQuery UI API - 插件桥（Widget Plugin Bridge）

所属类别

实用工具（Utilities） | 小部件（Widgets）

用法

描述：jQuery.widget.bridge() 方法是 jQuery 部件库（Widget Factory） 的一部分。它扮演着由 $.widget() 创建的对象和 jQuery API 之间的中介。

```
jQuery.widget.bridge( name, constructor )
```

参数	类型	类型
name	String	要创建的插件名称。
constructor	Function()	当插件被调用时要实例化的对象。

`$.widget.bridge()` 做如下事情：

- 连接一个常规的 JavaScript 构造函数到 jQuery API。
- 自动创建对象实例，并存储在元素的 `$.data` 缓存内。
- 允许调用公有方法。
- 防止调用私有方法。
- 防止在未初始化的对象上调用方法。
- 防止多个初始化。

jQuery UI 小部件使用 `$.widget( "foo.bar", {} );` 语法定义一个对象来创建。给出一个带有五个 `.foo` ， `$( ".foo" ).bar();` 的 DOM 结构将创建 "bar" 对象的五个实例。 `$.widget.bridge()` 基于 "bar" 对象和一个公共的 API 在库内工作。因此，您可以通过编写 `$( ".foo" ).bar()` 来创建实例，通过编写 `$( ".foo" ).bar( "baz" )` 来调用方法。

如果您只想一次性初始化并调用方法，那么您所传递给 `jQuery.widget.bridge()` 的对象可以很小：

```
var Highlighter = function( options, element ) {
  this.options = options;
  this.element = $( element );
  this._set( 800 );
};
Highlighter.prototype = {
  toggle: function() {
    this._set( this.element.css( "font-weight") === 400 ? 800 : 40(
  },
  _set: function(value) {
    this.element.css( "font-weight", value );
  }
};
```

在这里，您需要的只是一个构造函数，接收两个参数：

- `options` ：一个配置选项的对象
- `element` ：该实例在其上创建的 DOM 元素

然后您可以使用桥（bridge）把该对象作为一个 jQuery 插件，且可以在任意的 jQuery 对象上使用它：

```
// Hook up the plugin
$.widget.bridge( "colorToggle", Highlighter );

// Initialize it on divs
$( "div" ).colorToggle().click(function() {
  // Call the public method on click
  $( this ).colorToggle( "toggle" );
});
```

为了使用桥（bridge）的所有特性，您的对象原型需要有一个 `_init()` 方法。该方法在调用插件且实例已存在时调用。在这种情况下，您还需要有一个 `option()` 方法。该方法将会以选项作为第一个参数被调用。如果没有选项，则参数为一个空对象。如需了解 `option` 方法的使用，请查看 `$.Widget` 。

桥（bridge）有一个可选的属性，如果存在：如果对象原型有一个 `widgetFullName` 属性，则该属性将被作为存储和检索实例的键。否则，将使用 name 参数。

jQuery UI API 类别 - 小部件（Widgets）

小部件（Widgets）是功能丰富、有状态的插件，它有一个完整的生命周期，带有方法和事件。您可以查看 部件库（Widget Factory）文档 了解更多详情。

API	描述	也属于类别
折叠面板部件（Accordion Widget）	把一对标题和内容面板转换成折叠面板。	
自动完成部件（Autocomplete Widget）	自动完成功能根据用户输入值进行搜索和过滤，让用户快速找到并从预设值列表中选择。	
按钮部件（Button Widget）	可主题化的按钮和按钮集合。	
日期选择器部件（Datepicker Widget）	从弹出框或在线日历选择一个日期。	
对话框部件（Dialog Widget）	在一个交互覆盖层中打开内容。	
部件库（Widget Factory）	使用与所有 jQuery UI 小部件相同的抽象化来创建有状态的 jQuery 插件。	实用工具（Utilities）
插件桥（Widget Plugin Bridge）	jQuery.widget.bridge() 方法是 jQuery 部件库（Widget Factory）的一部分。它扮演着由 $.widget() 创建的对象和 jQuery API 之间的中介。	实用工具（Utilities）
菜单部件（Menu Widget）	带有鼠标和键盘交互的用于导航的可主题化菜单。	
进度条部件（Progressbar Widget）	显示一个确定的或不确定的进程状态。	
滑块部件（Slider Widget）	拖动手柄可以选择一个数值。	
旋转器部件（Spinner	通过向上/向下按钮和箭头键处理，为输入数	

标签页部件（Tabs Widget）	一种多面板的单内容区，每个面板与列表中的标题相关。
工具提示框部件（Tooltip Widget）	可自定义的、可主题化的工具提示框，替代原生的工具提示框。

jQuery UI API - 折叠面板部件（Accordion Widget）

所属类别

小部件（Widgets）

用法

描述：把一对标题和内容面板转换成折叠面板。

版本新增：1.0

折叠面板容器的标记需要一对标题和内容面板。

```
<div id="accordion">
  <h3>First header</h3>
  <div>First content panel</div>
  <h3>Second header</h3>
  <div>Second content panel</div>
</div>
```

折叠面板支持任意标记，但是每个内容面板必须是与其相关的头部面板的下一个同级。请查看 `header` 选项了解如何使用自定义的标记结构。

面板可以通过设置 `active` 选项以编程的方式激活。

键盘交互

当焦点在标题（header）上时，下面的键盘命令可用：

- UP/LEFT - 移动焦点到上一个标题（header）。如果在第一个标题（header）上，则移动焦点到最后一个标题（header）上。
- DOWN/RIGHT - 移动焦点到下一个标题（header）。如果在最后一个标题（header）上，则移动焦点到第一个标题（header）上。
- HOME - 移动焦点到第一个标题（header）上。
- END - 移动焦点到最后一个标题（header）上。
- SPACE/ENTER - 激活与获得焦点的标题（header）相关的面板（panel）。

当焦点在面板（panel）中时，下面的键盘命令可用：

- CTRL+UP：移动焦点到相关的标题（header）。

主题化

折叠面板部件（Accordion Widget）使用 jQuery UI CSS 框架 来定义它的外观和感观的样式。如果需要使用折叠面板指定的样式，则可以使用下面的 CSS class 名称：

- `ui-accordion`：折叠面板的外层容器。
    - `ui-accordion-header`：折叠面板的标题。如果标题包含 `icons`，则标题会另外有个 `ui-accordion-icons` class。
    - `ui-accordion-content`：折叠面板的内容面板。

依赖

- UI 核心（UI Core）
- 部件库（Widget Factory）
- 特效核心（Effects Core）（可选的；当与 `animate` 选项一起使用时）

附加说明

- 该部件要求一些功能性的 CSS，否则将无法工作。如果您创建了一个自定义的主题，请使用小部件指定的 CSS 文件作为起点。

快速导航

选项	
activeanimatecollapsibledisabledeventheaderheightStyleicons	destroydisablee

active

类型：Boolean 或 Integer

描述：当前打开哪一个面板。

支持多个类型：

- **Boolean**：设置 `active` 为 `false` 将折叠所有的面板。这要求 `collapsible` 选项必须为 `true`。
- **Integer**：激活打开的面板索引，以零为基础。负值则表示从最后一个面板后退选择面板。

代码实例：

初始化带有指定 `active` 选项的 accordion：

```
$( ".selector" ).accordion({ active: 2 });
```

在初始化后，获取或设置 `active` 选项：

```
// getter
var active = $( ".selector" ).accordion( "option", "active" );

// setter
$( ".selector" ).accordion( "option", "active", 2 );
```

默认值：0

animate

类型：Boolean 或 Number 或 String 或 Object

描述：是否使用动画改变面板，且如何使用动画改变面板。

支持多个类型：

- **Boolean**： `false` 值将禁用动画。
- **Number**：easing 默认的持续时间，以毫秒计。
- **String**：默认的持续时间要使用的 easing 名称。
- **Object**： `easing` 和 `duration` 属性的动画设置。

- 上面任意的选项都可以包含 `down` 属性。
- 当被激活的面板有一个比当前激活面板较低的指数时，发生 "Down" 动画。

代码实例：

初始化带有指定 `animate` 选项的 accordion：

```
$( ".selector" ).accordion({ animate: "bounceslide" });
```

在初始化后，获取或设置 `animate` 选项：

```
// getter
var animate = $( ".selector" ).accordion( "option", "animate" );

// setter
$( ".selector" ).accordion( "option", "animate", "bounceslide" );
```

默认值：{}

collapsible

类型：Boolean

描述：所有部分是否都可以马上关闭。允许折叠激活的部分。

代码实例：

初始化带有指定 `collapsible` 选项的 accordion：

```
$( ".selector" ).accordion({ collapsible: true });
```

在初始化后，获取或设置 `collapsible` 选项：

```
// getter
var collapsible = $( ".selector" ).accordion( "option", "collapsibl

// setter
$( ".selector" ).accordion( "option", "collapsible", true );
```

默认值：false

disabled

类型：Boolean

描述：如果设置为 `true`，则禁用该 accordion。

代码实例：

初始化带有指定 `disabled` 选项的 accordion：

```
$( ".selector" ).accordion({ disabled: true });
```

在初始化后，获取或设置 `disabled` 选项：

```
// getter
var disabled = $( ".selector" ).accordion( "option", "disabled" );

// setter
$( ".selector" ).accordion( "option", "disabled", true );
```

默认值：false

event

类型：String

描述：accordion 头部会作出反应的事件，用以激活相关的面板。可以指定多个事件，用空格间隔。

代码实例：

初始化带有指定 `event` 选项的 accordion：

```
$( ".selector" ).accordion({ event: "mouseover" });
```

在初始化后，获取或设置 `event` 选项：

```
// getter
var event = $( ".selector" ).accordion( "option", "event" );

// setter
$( ".selector" ).accordion( "option", "event", "mouseover" );
```

默认值："click"

header

类型：Selector

描述：标题元素的选择器，通过主要 accordion 元素上的 .find() 进行应用。内容面板必须是紧跟在与其相关的标题后的同级元素。

代码实例：

初始化带有指定 `header` 选项的 accordion：

```
$( ".selector" ).accordion({ header: "h3" });
```

在初始化后，获取或设置 `header` 选项：

```
// getter
var header = $( ".selector" ).accordion( "option", "header" );

// setter
$( ".selector" ).accordion( "option", "header", "h3" );
```

默认值："> li > :first-child,> :not(li):even"

heightStyle

类型：String

描述：控制 accordion 和每个面板的高度。可能的值：

- `"auto"` ：所有的面板将会被设置为最高的面板的高度。
- `"fill"` ：基于 accordion 的父元素的高度，扩展到可用的高度。
- `"content"` ：每个面板的高度取决于它的内容。

代码实例：

初始化带有指定 `heightStyle` 选项的 accordion：

```
$( ".selector" ).accordion({ heightStyle: "fill" });
```

在初始化后，获取或设置 `heightStyle` 选项：

```
// getter
var heightStyle = $( ".selector" ).accordion( "option", "heightSty

// setter
$( ".selector" ).accordion( "option", "heightStyle", "fill" );
```

默认值："auto"

icons

类型：Object

描述：标题要使用的图标，与 jQuery UI CSS 框架提供的图标（Icons） 匹配。设置为 `false` 则不显示图标。

- header （string，默认值："ui-icon-triangle-1-e"）
- activeHeader （string，默认值："ui-icon-triangle-1-s"）

代码实例：

初始化带有指定 `icons` 选项的 accordion：

```
$( ".selector" ).accordion({ icons: { "header": "ui-icon-plus", "ac
```

在初始化后，获取或设置 `icons` 选项：

```
// getter
var icons = $( ".selector" ).accordion( "option", "icons" );

// setter
$( ".selector" ).accordion( "option", "icons", { "header": "ui-icon
```

默认值：{ "header": "ui-icon-triangle-1-e", "activeHeader": "ui-icon-triangle-1-s" }

方法

destroy()

类型：jQuery (plugin only)

描述：完全移除 accordion 功能。这会把元素返回到它的预初始化状态。

- 该方法不接受任何参数。

代码实例：

调用 destroy 方法：

```
$( ".selector" ).accordion( "destroy" );
```

disable()

类型：jQuery (plugin only)

描述：禁用 accordion。

- 该方法不接受任何参数。

代码实例：

调用 disable 方法：

```
$( ".selector" ).accordion( "disable" );
```

enable()

类型：jQuery (plugin only)

描述：启用 accordion。

- 该方法不接受任何参数。

代码实例：

调用 enable 方法：

```
$( ".selector" ).accordion( "enable" );
```

option(optionName)

类型：Object

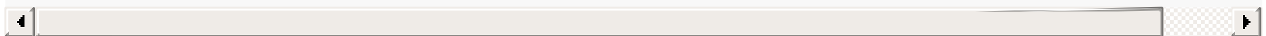
描述：获取当前与指定的 `optionName` 关联的值。

- **optionName** 类型：String 描述：要获取的选项的名称。

代码实例：

调用该方法：

```
var isDisabled = $( ".selector" ).accordion( "option", "disabled" 
```



option()

类型：PlainObject

描述：获取一个包含键/值对的对象，键/值对表示当前 accordion 选项哈希。

- 该方法不接受任何参数。

代码实例：

调用该方法：

```
var options = $( ".selector" ).accordion( "option" );
```

option(optionName, value)

类型：jQuery (plugin only)

描述：设置与指定的 `optionName` 关联的 accordion 选项的值。

- **optionName** 类型：String 描述：要设置的选项的名称。
- **value** 类型：Object 描述：要为选项设置的值。

代码实例：

调用该方法：

```
$( ".selector" ).accordion( "option", "disabled", true );
```

option(options)

类型：jQuery (plugin only)

描述：为 accordion 设置一个或多个选项。

- **options** 类型：Object 描述：要设置的 option-value 对。

代码实例：

调用该方法：

```
$( ".selector" ).accordion( "option", { disabled: true } );
```

refresh()

类型：jQuery (plugin only)

描述：处理任何在 DOM 中直接添加或移除的标题和面板，并重新计算 accordion 的高度。结果取决于内容和 `[heightStyle](#option-heightStyle )` 选项。

- 该方法不接受任何参数。

代码实例：

调用 refresh 方法：

```
$( ".selector" ).accordion( "refresh" );
```

widget()

类型：jQuery

描述：返回一个包含 accordion 的 `jQuery` 对象。

- 该方法不接受任何参数。

代码实例：

调用 widget 方法：

```
var widget = $( ".selector" ).accordion( "widget" );
```

事件

activate(event, ui)

类型：accordionactivate

描述：面板被激活后触发（在动画完成之后）。如果 accordion 之前是折叠的，则 `ui.oldHeader` 和 `ui.oldPanel` 将是空的 jQuery 对象。如果 accordion 正在折叠，则 `ui.newHeader` 和 `ui.newPanel` 将是空的 jQuery 对象。

注意：由于 **`activate`** 事件只有在面板激活时才能触发，当创建 **accordion** 部件时，最初的面板不会触发该事件。如果您需要一个用于部件创建的钩，请使用 **`create`** 事件。

- **event** 类型：Event
- **ui** 类型：Object
  - **newHeader** 类型：jQuery 描述：刚被激活的标题。
  - **oldHeader** 类型：jQuery 描述：刚被取消激活的标题。
  - **newPanel** 类型：jQuery 描述：刚被激活的面板。
  - **oldPanel** 类型：jQuery 描述：刚被取消激活的面板。

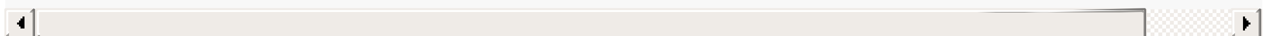
代码实例：

初始化带有指定 activate 回调的 accordion：

```
$( ".selector" ).accordion({
  activate: function( event, ui ) {}
});
```

绑定一个事件监听器到 accordionactivate 事件：

```
$( ".selector" ).on( "accordionactivate", function( event, ui ) {}
```



beforeActivate(event, ui)

类型：accordionbeforeactivate

描述：面板被激活前直接触发。可以取消以防止面板被激活。如果 accordion 当前是折叠的，则 `ui.oldHeader` 和 `ui.oldPanel` 将是空的 jQuery 对象。如果 accordion 正在折叠，则 `ui.newHeader` 和 `ui.newPanel` 将是空的 jQuery 对象。

- **event** 类型：Event
- **ui** 类型：Object
  - **newHeader** 类型：jQuery 描述：将被激活的标题。
  - **oldHeader** 类型：jQuery 描述：将被取消激活的标题。

    - **newPanel** 类型：jQuery 描述：将被激活的面板。
    - **oldPanel** 类型：jQuery 描述：将被取消激活的面板。

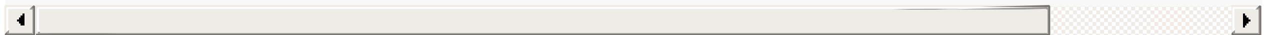
代码实例：

初始化带有指定 beforeActivate 回调的 accordion：

```
$( ".selector" ).accordion({
  beforeActivate: function( event, ui ) {}
});
```

绑定一个事件监听器到 accordionbeforeactivate 事件：

```
$( ".selector" ).on( "accordionbeforeactivate", function( event, ui
```



create(event, ui)

类型：accordioncreate

描述：当创建 accordion 时触发。如果 accordion 是折叠的，`ui.header` 和 `ui.panel` 将是空的 jQuery 对象。

- **event** 类型：Event
- **ui** 类型：Object
    - **header** 类型：jQuery 描述：激活的标题。
    - **panel** 类型：jQuery 描述：激活的面板。

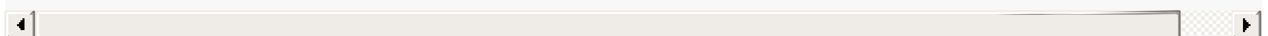
代码实例：

初始化带有指定 create 回调的 accordion：

```
$( ".selector" ).accordion({
  create: function( event, ui ) {}
});
```

绑定一个事件监听器到 accordioncreate 事件：

```
$( ".selector" ).on( "accordioncreate", function( event, ui ) {} );
```



实例

一个简单的 jQuery UI 折叠面板（Accordion）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>折叠面板部件（Accordion Widget）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<div id="accordion">
  <h3>部分 1</h3>
  <div>
    <p>Mauris mauris ante, blandit et, ultrices a, suscipit eget.
    Integer ut neque. Vivamus nisi metus, molestie vel, gravida in,
    condimentum sit amet, nunc. Nam a nibh. Donec suscipit eros.
    Nam mi. Proin viverra leo ut odio.</p>
  </div>
  <h3>部分 2</h3>
  <div>
    <p>Sed non urna. Phasellus eu ligula. Vestibulum sit amet purus
    Vivamus hendrerit, dolor aliquet laoreet, mauris turpis velit,
    faucibus interdum tellus libero ac justo.</p>
  </div>
  <h3>部分 3</h3>
  <div>
    <p>Nam enim risus, molestie et, porta ac, aliquam ac, risus.
    Quisque lobortis.Phasellus pellentesque purus in massa.</p>
    <ul>
      <li>List item one</li>
      <li>List item two</li>
      <li>List item three</li>
    </ul>
  </div>
</div>

<script>
$( "#accordion" ).accordion();
</script>

</body>
</html>
```

jQuery UI API - 自动完成部件（Autocomplete Widget）

所属类别

小部件（Widgets）

用法

描述：自动完成功能根据用户输入值进行搜索和过滤，让用户快速找到并从预设值列表中选择。

版本新增：1.8

任何可以接收输入的字段都可以转换为 Autocomplete，即，`<input>` 元素，`<textarea>` 元素及带有 `contenteditable` 属性的元素。

通过给 Autocomplete 字段焦点或者在其中输入字符，插件开始搜索匹配的条目并显示供选择的值的列表。通过输入更多的字符，用户可以过滤列表以获得更好的匹配。

该部件可用于选择先前选定的值，比如输入文章标签或者输入从地址簿中输入地址邮件地址。Autocomplete 也可以用来填充相关的信息，比如输入一个城市的名称来获得该城市的邮政编码。

您可以从本地源或者远程源获取数据：本地源适用于小数据集，例如，带有 50 个条目的地址簿；远程源适用于大数据集，比如，带有数百个或者成千上万个条目的数据库。如需了解更多有关自定义数据源的信息，请查看 `source` 选项的文档。

键盘交互

当菜单打开时，下面的键盘命令可用：

- UP - 移动焦点到上一个项目。如果在第一个项目上，则移动焦点到输入（input）。如果在输入（input）上，则移动焦点到最后一个项目。
- DOWN - 移动焦点到下一个项目。如果在最后一个项目上，则移动焦点到输入（input）。如果在输入（input）上，则移动焦点到第一个项目。
- ESCAPE - 关闭菜单。
- ENTER - 选择当前获得焦点的项目，并关闭菜单。
- TAB - 选择当前获得焦点的项目，关闭菜单，并移动焦点到下一个可聚焦（focusable）的元素。
- PAGE UP/DOWN - 滚动一屏的项目（基于菜单的高度）。

当菜单关闭时，下面的键盘命令可用：

- UP/DOWN - 如果满足 `minLength`，则打开菜单。

主题化

自动完成部件（Autocomplete Widget）使用 jQuery UI CSS 框架 来定义它的外观和感观的样式。如果需要使用自动完成部件指定的样式，则可以使用下面的 CSS class 名称：

- `ui-autocomplete`：用于显示匹配用户的 菜单（menu）
- `ui-autocomplete-input`：自动完成部件（Autocomplete Widget）实例化的 input 元素。

依赖

- UI 核心（UI Core）
- 部件库（Widget Factory）
- 定位（Position）
- 菜单部件（Menu Widget）

附加说明

- 该部件要求一些功能性的 CSS，否则将无法工作。如果您创建了一个自定义的主题，请使用小部件指定的 CSS 文件作为起点。
- 该部件以编程方式操作元素的值，因此当元素的值改变时不会触发原生的 `change` 事件。

快速导航

选项	
appendToautoFocusdelaydisabledminLengthpositionsource	closedestroydisab

appendTo

类型：Selector

描述：菜单应该被附加到哪一个元素。当该值为 `null` 时，输入域的父元素将检查 `ui-front` class。如果找到带有 `ui-front` class 的元素，菜单将被附加到该元素。如果未找到带有 `ui-front` class 的元素，不管值为多少，菜单将被附加到 body。

注意：当建议菜单打开时，**`appendTo`** 选项不应改变。

代码实例：

初始化带有指定 `appendTo` 选项的 autocomplete：

```
$( ".selector" ).autocomplete({ appendTo: "#someElem" });
```

在初始化后，获取或设置 `appendTo` 选项：

```
// getter
var appendTo = $( ".selector" ).autocomplete( "option", "appendTo"

// setter
$( ".selector" ).autocomplete( "option", "appendTo", "#someElem" );
```

默认值：null

autoFocus

类型：Boolean

描述：如果设置为 `true`，当菜单显示时，第一个条目将自动获得焦点。

代码实例：

初始化带有指定 `autoFocus` 选项的 autocomplete：

```
$( ".selector" ).autocomplete({ autoFocus: true });
```

在初始化后，获取或设置 `autoFocus` 选项：

```
// getter
var autoFocus = $( ".selector" ).autocomplete( "option", "autoFocus

// setter
$( ".selector" ).autocomplete( "option", "autoFocus", true );
```

默认值：false

delay

类型：Integer

描述：按键和执行搜索之间的延迟，以毫秒计。对于本地数据，采用零延迟是有意义的（更具响应性），但对于远程数据会产生大量的负荷，同时降低了响应性。

代码实例：

初始化带有指定 `delay` 选项的 autocomplete：

```
$( ".selector" ).autocomplete({ delay: 500 });
```

在初始化后，获取或设置 `delay` 选项：

```
// getter
var delay = $( ".selector" ).autocomplete( "option", "delay" );

// setter
$( ".selector" ).autocomplete( "option", "delay", 500 );
```

默认值：300

disabled

类型：Boolean

描述：如果设置为 `true`，则禁用该 autocomplete。

代码实例：

初始化带有指定 `disabled` 选项的 autocomplete：

```
$( ".selector" ).autocomplete({ disabled: true });
```

在初始化后，获取或设置 `disabled` 选项：

```
// getter
var disabled = $( ".selector" ).autocomplete( "option", "disabled"

// setter
$( ".selector" ).autocomplete( "option", "disabled", true );
```

默认值：false

minLength

类型：Integer

描述：执行搜索前用户必须输入的最小字符数。对于仅带有几项条目的本地数据，通常设置为零，但是当单个字符搜索会匹配几千项条目时，设置个高数值是很有必要的。

代码实例：

初始化带有指定 `minLength` 选项的 autocomplete：

```
$( ".selector" ).autocomplete({ minLength: 0 });
```

在初始化后，获取或设置 `minLength` 选项：

```
// getter
var minLength = $( ".selector" ).autocomplete( "option", "minLengtl

// setter
$( ".selector" ).autocomplete( "option", "minLength", 0 );
```

默认值：1

position

类型：Object

描述：标识建议菜单的位置与相关的 input 元素有关系。 `of` 选项默认为 input 元素，但是您可以指定另一个定位元素。如需了解各种选项的更多细节，请查看 jQuery UI 定位（Position）。

代码实例：

初始化带有指定 `position` 选项的 autocomplete：

```
$( ".selector" ).autocomplete({ position: { my : "right top", at: '
```

在初始化后，获取或设置 `position` 选项：

```
// getter
var position = $( ".selector" ).autocomplete( "option", "position"

// setter
$( ".selector" ).autocomplete( "option", "position", { my : "right
```

默认值：{ my: "left top", at: "left bottom", collision: "none" }

source

类型：Array 或 String 或 Function(Object request, Function response(Object data))

描述：定义要使用的数据，必须指定。

独立于您要使用的变量，标签总是被视为文本。如果您想要标签被视为 html，您可以使用 Scott González' html 扩展。演示侧重于 `source` 选项的不同变量 - 您可以查找其中匹配您的使用情况的那个，并查看相关代码。

支持多个类型：

- **Array**：可用于本地数据的一个数组。支持两种格式：
  - 字符串数组：`[ "Choice1", "Choice2" ]`
  - 带有 `label` 和 `value` 属性的对象数组：`[ { label: "Choice1", value: "value1" }, ... ]` label 属性显示在建议菜单中。当用户选择一个条目时，value 将被插入到 input 元素中。如果只是指定了一个属性，则该属性将被视为 label 和 value，例如，如果您只提供了 `value` 属性，value 也会被视为标签。
- **String**：当使用一个字符串，Autocomplete 插件希望该字符串指向一个能返回 JSON 数据的 URL 资源。它可以是在相同的主机上，也可以是在不同的主机上（必须提供 JSONP）。Autocomplete 插件不过滤结果，而是通过一个 `term` 字段添加了一个查询字符串，用于服务器端脚本过滤结果。例如，如果 `source` 选项设置为 `"http://example.com"`，且用户键入了 `foo`，GET 请求则为 `http://example.com?term=foo`。数据本身的格式可以与前面描述的本地数据的格式相同。
- **Function**：第三个变量，一个回调函数，提供最大的灵活性，可用于连接任何数据源到 Autocomplete。回调函数接受两个参数：
  - 一个 `request` 对象，带有一个 `term` 属性，表示当前文本输入中的值。例如，如果用户在 city 字段输入 `"new yo"`，则 Autocomplete term 等同于 `"new yo"`。
  - 一个 `response` 回调函数，提供单个参数：建议给用户的数据。该数据应基于被提供的 term 进行过滤，且可以是上面描述的本地数据的任何格式。用于在请求期间提供自定义 source 回调来处理错误。即使遇到错误，您也必须调用 `response` 回调函数。这就确保了小部件总是正确的状态。

  当过滤本地数据时，您可以使用内置的 `$.ui.autocomplete.escapeRegex` 函数。它会接受一个字符串参数，转义所有的正则表达式字符，让结果安全地传递到 `new RegExp()`。

代码实例：

初始化带有指定 `source` 选项的 autocomplete：

```
$( ".selector" ).autocomplete({ source: [ "c++", "java", "php", "co
```

在初始化后，获取或设置 `source` 选项：

```
// getter
var source = $( ".selector" ).autocomplete( "option", "source" );

// setter
$( ".selector" ).autocomplete( "option", "source", [ "c++", "java",
```

默认值：none; must be specified

方法

close()

类型：jQuery (plugin only)

描述：关闭 Autocomplete 菜单。当与 `[search](#method-search)` 方法结合使用时，可用于关闭打开的菜单。

- 该方法不接受任何参数。

代码实例：

调用 close 方法：

```
$( ".selector" ).autocomplete( "close" );
```

destroy()

类型：jQuery (plugin only)

描述：完全移除 autocomplete 功能。这会把元素返回到它的预初始化状态。

- 该方法不接受任何参数。

代码实例：

调用 destroy 方法：

```
$( ".selector" ).autocomplete( "destroy" );
```

disable()

类型：jQuery (plugin only)

描述：禁用 autocomplete。

- 该方法不接受任何参数。

代码实例：

调用 disable 方法：

```
$( ".selector" ).autocomplete( "disable" );
```

enable()

类型：jQuery (plugin only)

描述：启用 autocomplete。

- 该方法不接受任何参数。

代码实例：

调用 enable 方法：

```
$( ".selector" ).autocomplete( "enable" );
```

option(optionName)

类型：Object

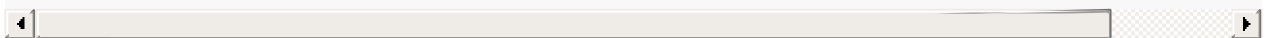
描述：获取当前与指定的 `optionName` 关联的值。

- **optionName** 类型：String 描述：要获取的选项的名称。

代码实例：

调用该方法：

```
var isDisabled = $( ".selector" ).autocomplete( "option", "disabled
```



option()

类型：PlainObject

描述：获取一个包含键/值对的对象，键/值对表示当前 autocomplete 选项哈希。

- 该方法不接受任何参数。

代码实例：

调用该方法：

```
var options = $( ".selector" ).autocomplete( "option" );
```

option(optionName, value)

类型：jQuery (plugin only)

描述：设置与指定的 `optionName` 关联的 autocomplete 选项的值。

- **optionName** 类型：String 描述：要设置的选项的名称。
- **value** 类型：Object 描述：要为选项设置的值。

代码实例：

调用该方法：

```
$( ".selector" ).autocomplete( "option", "disabled", true );
```

option(options)

类型：jQuery (plugin only)

描述：为 autocomplete 设置一个或多个选项。

- **options** 类型：Object 描述：要设置的 option-value 对。

代码实例：

调用该方法：

```
$( ".selector" ).autocomplete( "option", { disabled: true } );
```

search([value])

类型：jQuery (plugin only)

描述：触发 `search` 事件，如果该事件未被取消则调用数据源。当被点击时，可被类似选择框按钮用来打开建议。当不带参数调用该方法时，则使用当前输入的值。可带一个空字符串和 `minLength: 0` 进行调用，来显示所有的条目。

- **value** 类型：String

代码实例：

调用 search 方法：

```
$( ".selector" ).autocomplete( "search", "" );
```

widget()

类型：jQuery

描述：返回一个包含菜单元素的 `jQuery` 对象。虽然菜单项不断地被创建和销毁。菜单元素本身会在初始化时创建，并不断的重复使用。

- 该方法不接受任何参数。

代码实例：

调用 widget 方法：

```
$( ".selector" ).autocomplete( "widget" );
```

自动完成部件（Autocomplete Widget）通过 部件库（Widget Factory） 创建的，且可被扩展。当对部件进行扩展时，您可以重载或者添加扩展部件的行为。下面提供的方法作为扩展点，与上面列出的 插件方法 具有相同的 API 稳定性。如需了解更多有关小部件扩展的知识，请查看 通过部件库（Widget Factory）扩展小部件.

_renderItem(ul, item)

类型：jQuery

描述：Method that controls the creation of each option in the widget's menu. The method must create a new `<li>` element, append it to the menu, and return it.

Note: At this time the `<ul>` *element created must contain an* `<a>` *element for compatibility with the menu widget. See the example below.*

- **ul** 类型：jQuery 描述：新创建的 `<li>` 元素必须追加到的 `<ul>` 元素。
- **item** 类型：Object
    - **label** 类型：String 描述：条目显示的字符串。
    - **item** 类型：String 描述：当条目被选中时插入到输入框中的值。

代码实例：

添加条目的值作为 `<li>` 上的 data 属性。

```
_renderItem: function( ul, item ) {
  return $( "&lt;li&gt;" )
    .attr( "data-value", item.value )
    .append( $( "&lt;a&gt;" ).text( item.label ) )
    .appendTo( ul );
}
```

_renderMenu(ul, items)

类型：jQuery (plugin only)

描述：该方法负责在菜单显示前调整菜单尺寸。菜单元素可通过 `this.menu.element` 使用。

- **ul** 类型：jQuery 描述：一个要作为小部件的菜单使用的空的 `<ul>` 元素。
- **items** 类型：Array 描述：一个数组，数组元素为匹配用户输入的条目。每个条目是一个带有 `label` 和 `value` 属性的对象。

代码实例：

添加一个 CSS class 名称到旧的菜单项。

```
_renderMenu: function( ul, items ) {
  var that = this;
  $.each( items, function( index, item ) {
    that._renderItemData( ul, item );
  });
  $( ul ).find( "li:odd" ).addClass( "odd" );
}
```

_resizeMenu()

类型：jQuery (plugin only)

描述：该方法负责在菜单显示前调整菜单尺寸。菜单元素可通过 `this.menu.element` 使用。

- 该方法不接受任何参数。

代码实例：

菜单总是显示为 500 像素宽。

```
_resizeMenu: function() {
  this.menu.element.outerWidth( 500 );
}
```

事件

change(event, ui)

类型：autocompletechange

描述：如果输入域的值改变则触发该事件。

- **event** 类型：Event
- **ui** 类型：Object
  - **item** 类型：Object 描述：从菜单中选择的条目，否则属性为 `null` 。

代码实例：

初始化带有指定 change 回调的 autocomplete：

```
$( ".selector" ).autocomplete({
  change: function( event, ui ) {}
});
```

绑定一个事件监听器到 autocompletechange 事件：

```
$( ".selector" ).on( "autocompletechange", function( event, ui ) {}
```

close(event, ui)

类型：autocompleteclose

描述：当菜单隐藏时触发。不是每一个 `close` 事件都伴随着 `change` 事件。

- **event** 类型：Event
- **ui** 类型：Object

注意： `ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

代码实例：

初始化带有指定 close 回调的 autocomplete：

```
$( ".selector" ).autocomplete({
  close: function( event, ui ) {}
});
```

绑定一个事件监听器到 autocompleteclose 事件：

```
$( ".selector" ).on( "autocompleteclose", function( event, ui ) {}
```

create(event, ui)

类型：autocompletecreate

描述：当创建 autocomplete 时触发。

- **event** 类型：Event
- **ui** 类型：Object

注意：`ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

代码实例：

初始化带有指定 create 回调的 autocomplete：

```
$( ".selector" ).autocomplete({
  create: function( event, ui ) {}
});
```

绑定一个事件监听器到 autocompletecreate 事件：

```
$( ".selector" ).on( "autocompletecreate", function( event, ui ) {
```

focus(event, ui)

类型：autocompletefocus

描述：当焦点移动到一个条目上（未选择）时触发。默认的动作是把文本域中的值替换为获得焦点的条目的值，即使该事件是通过键盘交互触发的。取消该事件会阻止值被更新，但不会阻止菜单项获得焦点。

- **event** 类型：Event
- **ui** 类型：Object
  - **item** 类型：Object 描述：获得焦点的条目。

代码实例：

初始化带有指定 focus 回调的 autocomplete：

```
$( ".selector" ).autocomplete({
  focus: function( event, ui ) {}
});
```

绑定一个事件监听器到 autocompletefocus 事件：

```
$( ".selector" ).on( "autocompletefocus", function( event, ui ) {}
```

open(event, ui)

类型：autocompleteopen

描述：当打开建议菜单或者更新建议菜单时触发。

- **event** 类型：Event
- **ui** 类型：Object

注意：`ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

代码实例：

初始化带有指定 open 回调的 autocomplete：

```
$( ".selector" ).autocomplete({
  open: function( event, ui ) {}
});
```

绑定一个事件监听器到 autocompleteopen 事件：

```
$( ".selector" ).on( "autocompleteopen", function( event, ui ) {} 
```

response(event, ui)

类型：autocompleteresponse

描述：在搜索完成后菜单显示前触发。用于建议数据的本地操作，其中自定义的 `source` 选项回调不是必需的。该事件总是在搜索完成时触发，如果搜索无结果或者禁用了 Autocomplete，导致菜单未显示，该事件一样会被触发。

- **event** 类型：Event
- **ui** 类型：Object
  - **content** 类型：Array 描述：包含响应数据，且可被修改来改变显示结果。该数据已经标准化，所以如果您要修改数据，请确保每个条目都包含 `value` 和 `label` 属性。

代码实例：

初始化带有指定 response 回调的 autocomplete：

```
$( ".selector" ).autocomplete({
  response: function( event, ui ) {}
});
```

绑定一个事件监听器到 autocompleteresponse 事件：

```
$( ".selector" ).on( "autocompleteresponse", function( event, ui )
```

search(event, ui)

类型：autocompletesearch

描述：在搜索执行前满足 `minLength` 和 `delay` 后触发。如果取消该事件，则不会提交请求，也不会提供建议条目。

- **event** 类型：Event
- **ui** 类型：Object

注意：`ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

代码实例：

初始化带有指定 search 回调的 autocomplete：

```
$( ".selector" ).autocomplete({
  search: function( event, ui ) {}
});
```

绑定一个事件监听器到 autocompletesearch 事件：

```
$( ".selector" ).on( "autocompletesearch", function( event, ui ) {
```

select(event, ui)

类型：autocompleteselect

描述：当从菜单中选择条目时触发。默认的动作是把文本域中的值替换为被选中的条目的值。取消该事件会阻止值被更新，但不会阻止菜单关闭。

- **event** 类型：Event
- **ui** 类型：Object
  - **item** 类型：Object 描述：一个带有被选项的 `label` 和 `value` 属性的对象。

代码实例：

初始化带有指定 select 回调的 autocomplete：

```
$( ".selector" ).autocomplete({
  select: function( event, ui ) {}
});
```

绑定一个事件监听器到 autocompleteselect 事件：

```
$( ".selector" ).on( "autocompleteselect", function( event, ui ) {}
```

实例

实例 **1**：

一个简单的 jQuery UI 自动完成（Autocomplete）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>自动完成部件（Autocomplete Widget）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<label for="autocomplete">选择一个编程语言：</label>
<input id="autocomplete">

<script>
$( "#autocomplete" ).autocomplete({
  source: [ "c++", "java", "php", "coldfusion", "javascript", "asp'
});
</script>

</body>
</html>
```

实例 **2**：

使用自定义源回调来匹配条件的开始。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>自动完成部件（Autocomplete Widget）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<label for="autocomplete">选择一个编程语言：</label>
<input id="autocomplete">

<script>
var tags = [ "c++", "java", "php", "coldfusion", "javascript", "asp
$( "#autocomplete" ).autocomplete({
  source: function( request, response ) {
          var matcher = new RegExp( "^" + $.ui.autocomplete.escapeR
          response( $.grep( tags, function( item ){
              return matcher.test( item );
          }) );
      }
});
</script>

</body>
</html>
```

jQuery UI API - 按钮部件（Button Widget）

所属类别

小部件（Widgets）

用法

描述：可主题化的按钮和按钮集合。

版本新增：1.8

按钮部件（Button Widget）加强了标准表单元素的功能，比如按钮（button）、输入（input）、锚（anchor），用适当的悬停（hover）和激活（active）样式来主题化按钮。

除了基本的按钮，单选按钮和复选框（input 类型为 radio 和 checkbox）也可以转换为按钮。相关的标签（label）设计成按钮的样式，点击时更新底层的输入。为了能正常工作，需要给 input 一个 `id` 属性，并指向标签（label）的 `for` 属性。不要把 input 放在标签（label）内，否则会引起可访问性问题。

为了分组单选按钮，Button 也提供了一个额外的小部件，名为 Buttonset。Buttonset 通过选择一个容器元素（包含单选按钮）并调用 `.buttonset()` 来使用。Buttonset 也提供了可视化分组，因此当有一组按钮时都可考虑使用它。它会选择所有的后代，并对它们应用 `.button()`。您可以启用和禁用一个按钮集，这将会启用和禁用所有包含的按钮。销毁按钮集会调用每个按钮的 `destroy` 方法。对于分组的单选按钮和复选框按钮，推荐使用带有 `legend` 的 `fieldset` 来提供一个可访问的分组标签。

当使用一个类型为 button、submit 或 reset 的 input 时，仅限于支持纯文本无图标标签。

主题化

按钮部件（Button Widget）使用 jQuery UI CSS 框架 来定义它的外观和感观的样式。如果需要使用按钮指定的样式，则可以使用下面的 CSS class 名称：

- `ui-button`：表示按钮的 DOM 元素。该元素会根据 text 和 icons 选项添加下列 class 之一：`ui-button-text-only`、`ui-button-icon-only`、`ui-button-icon`、`ui-button-text-icons`。
  - `ui-button-icon-primary`：用于显示按钮主要图标的元素。只有当主要图标在 icons 选项中提供时才呈现。
  - `ui-button-text`：在按钮的文本内容周围的容器。

    - `ui-button-icon-secondary`：用于显示按钮的次要图标。只有当次要图标在 icons 选项中提供时才呈现。
- `ui-buttonset`：Buttonset 的外层容器。

依赖

- UI 核心（UI Core）
- 部件库（Widget Factory）

附加说明

- 该部件要求一些功能性的 CSS，否则将无法工作。如果您创建了一个自定义的主题，请使用小部件指定的 CSS 文件作为起点。

快速导航

选项	方法	事件
disablediconslabeltext	destroydisableenableoptionrefreshwidget	create

disabled

类型：Boolean

描述：如果设置为 `true`，则禁用该 button。

代码实例：

初始化带有指定 `disabled` 选项的 button：

```
$( ".selector" ).button({ disabled: true });
```

在初始化后，获取或设置 `disabled` 选项：

```
// getter
var disabled = $( ".selector" ).button( "option", "disabled" );

// setter
$( ".selector" ).button( "option", "disabled", true );
```

默认值：false

icons

类型：Object

描述：要显示的图标，包括带有文本的图标和不带有文本的图标（查看 `text` 选项）。默认情况下，主图标显示在标签文本的左边，副图标显示在右边。显示位置可通过 CSS 进行控制。

`primary` 和 `secondary` 属性值必须是 图标 class 名称，例如，`"ui-icon-gear"`。如果只使用一个图标，则 `icons: { primary: "ui-icon-locked" }`。如果使用两个图标，则 `icons: { primary: "ui-icon-gear", secondary: "ui-icon-triangle-1-s"`。

代码实例：

初始化带有指定 `icons` 选项的 button：

```
$( ".selector" ).button({ icons: { primary: "ui-icon-gear", seconda
```

在初始化后，获取或设置 `disabled` 选项：

```
// getter
var icons = $( ".selector" ).button( "option", "icons" );

// setter
$( ".selector" ).button( "option", "icons", { primary: "ui-icon-gea
```

默认值：{ primary: null, secondary: null }

label

类型：String

描述：要显示在按钮中的文本。当未指定时（`null`），则使用元素的 HTML 内容，或者如果元素是一个 submit 或 reset 类型的 input 元素，则使用它的 `value` 属性，或者如果元素是一个 radio 或 checkbox 类型的 input 元素，则使用相关的 label 元素的 HTML 内容。

代码实例：

初始化带有指定 `label` 选项的 button：

```
$( ".selector" ).button({ label: "custom label" });
```

在初始化后，获取或设置 `label` 选项：

```
// getter
var label = $( ".selector" ).button( "option", "label" );

// setter
$( ".selector" ).button( "option", "label", "custom label" );
```

默认值：null

text

类型：Boolean

描述：是否显示标签。当设置为 `false` 时，不显示文本，但是此时必须启用 `icons` 选项，否则 `text` 选项将被忽略。

代码实例：

初始化带有指定 `text` 选项的 button：

```
$( ".selector" ).button({ text: false });
```

在初始化后，获取或设置 `text` 选项：

```
// getter
var text = $( ".selector" ).button( "option", "text" );

// setter
$( ".selector" ).button( "option", "text", false );
```

默认值：true

方法

destroy()

类型：jQuery (plugin only)

描述：完全移除 button 功能。这会把元素返回到它的预初始化状态。

- 该方法不接受任何参数。

代码实例：

调用 destroy 方法：

```
$( ".selector" ).button( "destroy" );
```

disable()

类型：jQuery (plugin only)

描述：禁用 button。

- 该方法不接受任何参数。

代码实例：

调用 disable 方法：

```
$( ".selector" ).button( "disable" );
```

enable()

类型：jQuery (plugin only)

描述：启用 button。

- 该方法不接受任何参数。

代码实例：

调用 enable 方法：

```
$( ".selector" ).button( "enable" );
```

option(optionName)

类型：Object

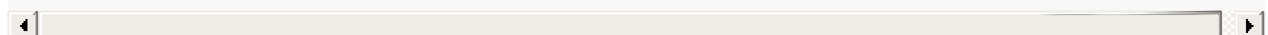
描述：获取当前与指定的 `optionName` 关联的值。

- **optionName** 类型：String 描述：要获取的选项的名称。

代码实例：

调用该方法：

```
var isDisabled = $( ".selector" ).button( "option", "disabled" );
```



option()

类型：PlainObject

描述：获取一个包含键/值对的对象，键/值对表示当前 button 选项哈希。

- 该方法不接受任何参数。

代码实例：

调用该方法：

```
var options = $( ".selector" ).button( "option" );
```

option(optionName, value)

类型：jQuery (plugin only)

描述：设置与指定的 `optionName` 关联的 button 选项的值。

- **optionName** 类型：String 描述：要设置的选项的名称。
- **value** 类型：Object 描述：要为选项设置的值。

代码实例：

调用该方法：

```
$( ".selector" ).button( "option", "disabled", true );
```

option(options)

类型：jQuery (plugin only)

描述：为 button 设置一个或多个选项。

- **options** 类型：Object 描述：要设置的 option-value 对。

代码实例：

调用该方法：

```
$( ".selector" ).button( "option", { disabled: true } );
```

refresh()

类型：jQuery (plugin only)

描述：刷新按钮的视觉状态。用于在以编程方式改变原生元素的选中状态或禁用状态后更新按钮状态。

- 该方法不接受任何参数。

代码实例：

调用 refresh 方法：

```
$( ".selector" ).button( "refresh" );
```

widget()

类型：jQuery

描述：返回一个包含 button 的 `jQuery` 对象。

- 该方法不接受任何参数。

代码实例：

调用 widget 方法：

```
var widget = $( ".selector" ).button( "widget" );
```

事件

create(event, ui)

类型：buttoncreate

描述：当创建按钮 button 时触发。

- **event** 类型：Event
- **ui** 类型：Object

注意：`ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

代码实例：

初始化带有指定 create 回调的 button：

```
$( ".selector" ).button({
  create: function( event, ui ) {}
});
```

绑定一个事件监听器到 buttoncreate 事件：

```
$( ".selector" ).on( "buttoncreate", function( event, ui ) {} );
```

实例

实例 **1**：

一个简单的 jQuery UI 按钮（Button）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>按钮部件（Button Widget）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<button>按钮标签</button>

<script>
$( "button" ).button();
</script>

</body>
</html>
```

实例 **2**：

一个简单的 jQuery UI 按钮集（Buttonset）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>按钮部件（Button Widget）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<form>
  <fieldset>
    <legend>最喜欢的 jQuery 项目</legend>
    <div id="radio">
      <input type="radio" id="sizzle" name="project">
      <label for="sizzle">Sizzle</label>

      <input type="radio" id="qunit" name="project" checked="checke
      <label for="qunit">QUnit</label>

      <input type="radio" id="color" name="project">
      <label for="color">Color</label>
    </div>
  </fieldset>
</form>

<script>
$( "#radio" ).buttonset();
</script>

</body>
</html>
```

jQuery UI API - 日期选择器部件（Datepicker Widget）

所属类别

小部件（Widgets）

用法

描述：从弹出框或内联日历选择一个日期。

版本新增：1.0

jQuery UI 日期选择器（Datepicker）是向页面添加日期选择功能的高度可配置插件。您可以自定义日期格式和语言，限制可选择的日期范围，添加按钮和其他导航选项。

默认情况下，当相关的文本域获得焦点时，在一个小的覆盖层打开日期选择器。对于一个内联的日历，只需简单地将日期选择器附加到 div 或者 span 上。

键盘交互

当日期选择器打开时，下面的键盘命令可用：

- PAGE UP：移到上一个月。
- PAGE DOWN：移到下一个月。
- CTRL+PAGE UP：移到上一年。
- CTRL+PAGE DOWN：移到下一年。
- CTRL+HOME：移到当前月份。如果日期选择器是关闭的则打开。
- CTRL+LEFT：移到上一天。
- CTRL+RIGHT：移到下一天。
- CTRL+UP：移到上一周。
- CTRL+DOWN：移到下一周。
- ENTER：选择聚焦的日期。
- CTRL+END：关闭日期选择器，并清除日期。
- ESCAPE：关闭日期选择器，不做任何选择。

实用功能

$.datepicker.setDefaults(settings)

为所有的日期选择器改变默认设置。

使用 `option()` 方法来改变个别实例的设置。

代码实例：

设置所有的日期选择器在获得焦点时或点击图标时打开。

```
$.datepicker.setDefaults({
  showOn: "both",
  buttonImageOnly: true,
  buttonImage: "calendar.gif",
  buttonText: "Calendar"
});
```

设置所有的日期选择器都有法语文本。

```
$.datepicker.setDefaults( $.datepicker.regional[ "fr" ] );
```

$.datepicker.formatDate(format, date, settings)

格式化日期为一个带有指定格式的字符串值。

格式可以是下列组合：

- d - 一月中的第几天（没有前导零）
- dd - 一月中的第几天（两位数）
- o - 一年中的第几天（没有前导零）
- oo - 一年中的第几天（三位数）
- D - 天的短名称
- DD - 天的长名称
- m - 一年中的第几月（没有前导零）
- mm - 一年中的第几月（两位数）
- M - 月的短名称
- MM - 月的长名称
- y - 年（两位数）
- yy - 年（四位数）
- @ - Unix 时间戳（ms since 01/01/1970）
- ! - Windows 钟表（100ns since 01/01/0001）
- '...' - 文本
- '' - 单引号
- 其他 - 文本

也有一些 `$.datepicker` 预定义的标准日期格式：

- ATOM - 'yy-mm-dd' （与 RFC 3339/ISO 8601 相同）
- COOKIE - 'D, dd M yy'
- ISO_8601 - 'yy-mm-dd'
- RFC_822 - 'D, d M y' （参照 RFC 822）
- RFC_850 - 'DD, dd-M-y' （参照 RFC 850）

- RFC_1036 - 'D, d M y' （参照 RFC 1036）
- RFC_1123 - 'D, d M yy' （参照 RFC 1123）
- RFC_2822 - 'D, d M yy' （参照 RFC 2822）
- RSS - 'D, d M y' （与 RFC 822 相同）
- TICKS - '!'
- TIMESTAMP - '@'
- W3C - 'yy-mm-dd' （与 ISO 8601 相同）

代码实例：

以 ISO 格式显示日期。产生 "2007-01-26"。

```
$.datepicker.formatDate( "yy-mm-dd", new Date( 2007, 1 - 1, 26 ) );
```

以扩展法语格式显示日期。产生 "Samedi, Juillet 14, 2007"。

```
$.datepicker.formatDate( "DD, MM d, yy", new Date( 2007, 7 - 1, 14
  dayNamesShort: $.datepicker.regional[ "fr" ].dayNamesShort,
  dayNames: $.datepicker.regional[ "fr" ].dayNames,
  monthNamesShort: $.datepicker.regional[ "fr" ].monthNamesShort,
  monthNames: $.datepicker.regional[ "fr" ].monthNames
});
```

$.datepicker.parseDate(format, value, settings)

从一个指定格式的字符串值中提取日期。

格式可以是下列组合：

- d - 一月中的第几天（没有前导零）
- dd - 一月中的第几天（两位数）
- o - 一年中的第几天（没有前导零）
- oo - 一年中的第几天（三位数）
- D - 星期几的短名称
- DD - 星期几的长名称
- m - 一年中的第几月（没有前导零）
- mm - 一年中的第几月（两位数）
- M - 月的短名称
- MM - 月的长名称
- y - 年（两位数）
- yy - 年（四位数）
- @ - Unix 时间戳（ms since 01/01/1970）
- ! - Windows 钟表（100ns since 01/01/0001）
- '...' - 文本
- '' - 单引号

- 其他 - 文本

一些可能被抛出的异常：

- 'Invalid arguments' - 如果格式或值为空则抛出此异常。
- 'Missing number at position nn' - 如果格式显示一个未找到的数值则抛出此异常。
- 'Unknown name at position nn' - 如果格式显示一个未找到的星期几名称或月份名称则抛出此异常。
- 'Unexpected literal at position nn' - 如果格式显示一个未找到的文本值则抛出此异常。
- 'Invalid date' - 如果日期无效则抛出此异常，比如 '31/02/2007'。

代码实例：

提取一个 ISO 格式的日期。

```
$.datepicker.parseDate( "yy-mm-dd", "2007-01-26" );
```

提取一个扩展法语格式的日期。

```
$.datepicker.parseDate( "DD, MM d, yy", "Samedi, Juillet 14, 2007",
  shortYearCuroff: 20,
  dayNamesShort: $.datepicker.regional[ "fr" ].dayNamesShort,
  dayNames: $.datepicker.regional[ "fr" ].dayNames,
  monthNamesShort: $.datepicker.regional[ "fr" ].monthNamesShort,
  monthNames: $.datepicker.regional[ "fr" ].monthNames
});
```

$.datepicker.iso8601Week(date)

确定一个给定的日期在一年中的第几周：1 到 53。

该函数使用 ISO 8601 定义一周：一周从星期一开始，每一年的第一周包含 1 月 4 日。这意味着上一年至多有三天可能包含在当年的第一周中，当年至多有三天可能包含在上一年的最后一周中。

该函数是 `calculateWeek` 选项的默认实现。

代码实例：

查找日期在一年中的第几周。

```
$.datepicker.iso8601Week( new Date( 2007, 1 - 1, 26 ) );
```

$.datepicker.noWeekends

设置如 beforeShowDay 函数，防止选择周末。

我们可以在 `beforeShowDay` 选项中提供 `noWeekends()` 函数，用来计算所有工作日，提供一个 `true / false` 值的数组，用来指示日期是否可选择。

代码实例：

设置 DatePicker，让周末不可选。

```
$( "#datepicker" ).datepicker({
  beforeShowDay: $.datepicker.noWeekends
});
```

局限

日期选择器提供了迎合不同的语言和日期格式本地化内容的支持。每个本地化包含在名称后追加语言代码的文件中，例如法语为 `jquery.ui.datepicker-fr.js`。所需的本地化文件需要包含在主要的日期选择器代码后面。每个本地化文件添加了它自己的设置到可用的本地化集合中，所有实例自动应用这些设置为默认设置。

`$.datepicker.regional` 属性保存了一个本地化数组，以语言代码作为前置，默认前置为 `""`，表示英语。每个条目是一个带有下列属性的对象：`closeText`、`prevText`、`nextText`、`currentText`、`monthNames`、`monthNamesShort`、`dayNames`、`dayNamesShort`、`dayNamesMin`、`weekH`、`dateFormat`、`firstDay`、`isRTL`、`showMonthAfterYear` 和 `yearSuffix`。

您可以通过下面代码恢复默认的本地化：

`$.datepicker.setDefaults( $.datepicker.regional[ "" ] );`

您可以通过下面代码覆盖一个特定地点的日期选择器：

`$( selector ).datepicker( $.datepicker.regional[ "fr" ] );`

主题化

日期选择器部件（Datepicker Widget）使用 jQuery UI CSS 框架 来定义它的外观和感观的样式。如果需要使用日期选择器指定的样式，则可以使用下面的 CSS class 名称：

- `ui-datepicker`：日期选择器的外层容器。如果日期选择器是内联的，该元素会另外带有一个 `ui-datepicker-inline` class。如果设置了 `isRTL` 选项，该元素会另外带有一个 `ui-datepicker-rtl` class。
  - `ui-datepicker-header`：日期选择器的头部容器。
    - `ui-datepicker-prev`：用于选择上一月的控件。
    - `ui-datepicker-next`：用于选择下一月的控件。
    - `ui-datepicker-title`：日期选择器包含月和年的标题容器。

        - `ui-datepicker-month`：月的文本显示，如果设置了 `changeMonth` 选项则显示 `<select>` 元素。
        - `ui-datepicker-year`：年的文本显示，如果设置了 `changeYear` 选项则显示 `<select>` 元素。
    - `ui-datepicker-calendar`：包含日历的表格。
        - `ui-datepicker-week-end`：周末的单元格。: Cells containing weekend days.
        - `ui-datepicker-other-month`：发生在某月但不是当前月天数的单元格。
        - `ui-datepicker-unselectable`：用户不可选择的单元格。
        - `ui-datepicker-current-day`：已选中日期的单元格。
        - `ui-datepicker-today`：当天日期的单元格。
    - `ui-datepicker-buttonpane`：当设置 `showButtonPanel` 选项时使用按钮面板（buttonpane）。
        - `ui-datepicker-current`：用于选择当天日期的按钮。

如果 `numberOfMonths` 选项用于显示多个月份，则使用一些额外的 class：

- `ui-datepicker-multi`：一个多月份日期选择器的最外层容器。该元素会根据要显示的月份个数另外带有 `ui-datepicker-multi-2`、`ui-datepicker-multi-3` 或 `ui-datepicker-multi-4` class 名称。
    - `ui-datepicker-group`：分组内单独的选择器。该元素会根据它在分组中的位置另外带有 `ui-datepicker-group-first`、`ui-datepicker-group-middle` 或 `ui-datepicker-group-last` class 名称。

依赖

- UI 核心（UI Core）
- 特效核心（Effects Core）（可选的；当与 `showAnim` 选项一起使用时）

附加说明

- 该部件要求一些功能性的 CSS，否则将无法工作。如果您创建了一个自定义的主题，请使用小部件指定的 CSS 文件作为起点。
- 该部件以编程方式操作元素的值，因此当元素的值改变时不会触发原生的 `change` 事件。
- 不支持在 `<input type="date">` 上创建日期选择器，因为会造成与本地选择器的 UI 冲突。

实例

一个简单的 jQuery UI 日期选择器（Datepicker）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>日期选择器部件（Datepicker Widget）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<div id="datepicker"></div>

<script>
$( "#datepicker" ).datepicker();
</script>

</body>
</html>
```

jQuery UI API - 对话框部件（Dialog Widget）

所属类别

小部件（Widgets）

用法

描述：在一个交互覆盖层中打开内容。

版本新增：1.0

对话框是一个悬浮窗口，包括一个标题栏和一个内容区域。对话框窗口可以移动，重新调整大小，默认情况下通过 'x' 图标关闭。

如果内容长度超过最大高度，一个滚动条会自动出现。

一个底部按钮栏和一个半透明的模式覆盖层是常见的添加选项。

焦点

当打开一个对话框时，焦点会自动移动到满足下面条件的第一个项目：

1. 带有 `autofocus` 属性的对话框内的第一个元素
2. 对话框内容内的第一个 `:tabbable` 元素
3. 对话框按钮面板内的第一个 `:tabbable` 元素
4. 对话框的关闭按钮
5. 对话框本身

当打开时，对话框部件（Dialog Widget）确保通过 tab 切换对话框内元素间的焦点，不包括对话框外的元素。模态对话框防止鼠标用户点击对话框外的元素。

当关闭对话框时，焦点自动返回到之前对话框打开时获得焦点的元素上。

隐藏关闭按钮

在一些情况下，您可能想要隐藏关闭按钮，例如，在按钮面板中已经有一个关闭按钮的情况。最好的解决方法是通过 CSS。作为实例，您可以定义一个简单的规则，比如：

```
.no-close .ui-dialog-titlebar-close {
  display: none;
}
```

然后，您可以添加 `no-close` class 到任意的对话框，用来隐藏关闭按钮：

```
$( "#dialog" ).dialog({
  dialogClass: "no-close",
  buttons: [
    {
      text: "OK",
      click: function() {
        $( this ).dialog( "close" );
      }
    }
  ]
});
```

主题化

对话框部件（Dialog Widget）使用 jQuery UI CSS 框架 来定义它的外观和感观的样式。如果需要使用对话框指定的样式，则可以使用下面的 CSS class 名称：

- `ui-dialog` ：对话框的外层容器。
    - `ui-dialog-titlebar` ：包含对话框标题和关闭按钮的标题栏。
        - `ui-dialog-title` ：对话框文本标题周围的容器。
        - `ui-dialog-titlebar-close` ：对话框的关闭按钮。
    - `ui-dialog-content` ：对话框内容周围的容器。这也是部件被实例化的元素。
    - `ui-dialog-buttonpane` ：包含对话按钮的面板。只有当设置了 `buttons` 选项时才呈现。
        - `ui-dialog-buttonset` ：按钮周围的容器。

此外，当设置了 `modal` 选项时，一个带有 `ui-widget-overlay` class 名称的元素被追加到 `<body>` 。

依赖

- UI 核心（UI Core）
- 部件库（Widget Factory）
- 定位（Position）
- 按钮部件（Button Widget）
- 可拖拽小部件（Draggable Widget）（可选的；当与 `draggable` 选项一起使用时）
- 可调整尺寸小部件（Resizable Widget）（可选的；当与 `resizable` 选项一起使用时）
- 特效核心（Effects Core）（可选的；当与 `show` 和 `hide` 选项一起使用时）

附加说明

- 该部件要求一些功能性的 CSS，否则将无法工作。如果您创建了一个自定义的主题，请使用小部件指定的 CSS 文件作为起点。

实例

一个简单的 jQuery UI 对话框（Dialog）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>对话框部件（Dialog Widget）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<button id="opener">打开对话框</button>
<div id="dialog" title="对话框标题">我是一个对话框</div>

<script>
$( "#dialog" ).dialog({ autoOpen: false });
$( "#opener" ).click(function() {
  $( "#dialog" ).dialog( "open" );
});
</script>

</body>
</html>
```

jQuery UI API - 菜单部件（Menu Widget）

所属类别

小部件（Widgets）

用法

描述：带有鼠标和键盘交互的用于导航的可主题化菜单。

版本新增：1.9

菜单可以用任何有效的标记创建，只要元素有严格的父/子关系且每个条目都有一个锚。最常用的元素是无序列表（ `<ul>` ）：

```
<ul id="menu">
  <li><a href="#">Item 1</a></li>
  <li><a href="#">Item 2</a></li>
  <li><a href="#">Item 3</a>
    <ul>
      <li><a href="#">Item 3-1</a></li>
      <li><a href="#">Item 3-2</a></li>
      <li><a href="#">Item 3-3</a></li>
      <li><a href="#">Item 3-4</a></li>
      <li><a href="#">Item 3-5</a></li>
    </ul>
  </li>
  <li><a href="#">Item 4</a></li>
  <li><a href="#">Item 5</a></li>
</ul>
```

如果使用一个非 `<ul>` / `<li>` 的结构，为菜单和菜单条目使用相同的元素，请使用 `menus` 选项来区分两个元素，例如 `menus: "div.menuElement"` 。

可通过向元素添加 `ui-state-disabled` class 来禁用任何菜单条目。

图标

为了向菜单添加图标，请在标记中包含图标：

```
<ul id="menu">
  <li><a href="#"><span class="ui-icon ui-icon-disk"></span>Save</a
</ul>
```

菜单（Menu）会自动向无图标的条目添加必要的内边距。

分隔符

分隔符元素可通过包含未链接的菜单条目来创建，菜单条目只能是空格/破折号：

```
<ul id="menu">
  <li><a href="#">Item 1</a></li>
  <li>-</li>
  <li><a href="#">Item 2</a></li>
</ul>
```

键盘交互

- ENTER/SPACE：调用获得焦点的菜单项的动作，可能会打开一个子菜单。
- UP：移动教导到上一个菜单项。
- DOWN：移动教导到下一个菜单项。
- RIGHT：如果可用，则打开子菜单。
- LEFT：关闭当前子菜单，移动焦点到父菜单项。如果焦点不在子菜单上，则不进行任何操作。
- ESCAPE：关闭当前子菜单，移动焦点到父菜单项。如果焦点不在子菜单上，则不进行任何操作。

输入一个字母，移动焦点到以该字母开头的第一个条目。重复相同的字符会循环显示匹配的条目。在一个时间内输入更多的字符则匹配所输入的字符。

禁用项可获得键盘焦点，但是不允许任何交互。

主题化

菜单部件（Menu Widget）使用 jQuery UI CSS 框架 来定义它的外观和感观的样式。如果需要使用菜单指定的样式，则可以使用下面的 CSS class 名称：

- `ui-menu`：菜单的外层容器。如果菜单包含图标，该元素会另外带有一个 `ui-menu-icons` class。
  - `ui-menu-item`：单个菜单项的容器。
    - `ui-menu-icon`：通过 `icons` 选项进行子菜单图标设置。
  - `ui-menu-divider`：菜单项之间的分隔符元素。

依赖

- UI 核心（UI Core）
- 部件库（Widget Factory）
- 定位（Position）

附加说明

- 该部件要求一些功能性的 CSS，否则将无法工作。如果您创建了一个自定义的主题，请使用小部件指定的 CSS 文件作为起点。

实例

一个简单的 jQuery UI 菜单（Menu）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>菜单部件（Menu Widget）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  .ui-menu {
    width: 200px;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<ul id="menu">
  <li><a href="#">Item 1</a></li>
  <li><a href="#">Item 2</a></li>
  <li><a href="#">Item 3</a>
    <ul>
      <li><a href="#">Item 3-1</a></li>
      <li><a href="#">Item 3-2</a></li>
      <li><a href="#">Item 3-3</a></li>
      <li><a href="#">Item 3-4</a></li>
      <li><a href="#">Item 3-5</a></li>
    </ul>
  </li>
  <li><a href="#">Item 4</a></li>
  <li><a href="#">Item 5</a></li>
</ul>

<script>
$( "#menu" ).menu();
</script>

</body>
</html>
```

jQuery UI API - 进度条部件（Progressbar Widget）

所属类别

小部件（Widgets）

用法

描述：显示一个确定的或不确定的进程状态。

版本新增：1.6

进度条被设计来显示进度的当前完成百分比。进度条通过 CSS 编码灵活调整大小，默认会缩放到适应父容器的大小。

一个确定的进度条只能在系统可以准确更新当前状态的情况下使用。一个确定的进度条不会从左向右填充，然后循环回到空 - 如果不能计算实际状态，则使用不确定的进度条以便提供用户反馈。

主题化

进度条部件（Progressbar Widget）使用 jQuery UI CSS 框架 来定义它的外观和感观的样式。如果需要使用进度条指定的样式，则可以使用下面的 CSS class 名称：

- `ui-progressbar`：进度条的外层容器。该元素会为不确定的进度条另外添加一个 `ui-progressbar-indeterminate` class。
    - `ui-progressbar-value`：该元素代表进度条的填充部分。
        - `ui-progressbar-overlay`：用于为不确定的进度条显示动画的覆盖层。

依赖

- UI 核心（UI Core）
- 部件库（Widget Factory）

附加说明

- 该部件要求一些功能性的 CSS，否则将无法工作。如果您创建了一个自定义的主题，请使用小部件指定的 CSS 文件作为起点。

实例

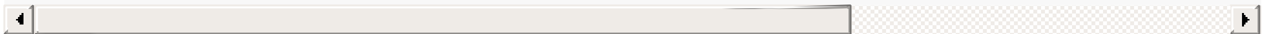
一个简单的 jQuery UI 不确定的进度条（Indeterminate Progressbar）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>进度条部件（Progressbar Widget）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<div id="progressbar"></div>

<script>
$( "#progressbar" ).progressbar({
  value: false
});
</script>

</body>
</html>
```



jQuery UI API - 滑块部件（Slider Widget）

所属类别

小部件（Widgets）

用法

描述：拖动手柄来选择一个数值。

版本新增：1.5

jQuery UI 滑块（Slider）插件允许通过滑块进行选择。有各种不同的选项，比如多个手柄和范围。手柄可通过鼠标或箭头按键进行移动。

滑块部件（Slider Widget）会在初始化时创建带有 class `ui-slider-handle` 的手柄元素。您可以通过在初始化之前创建并追加元素，同时向元素添加 `ui-slider-handle` class 来指定自定义的手柄元素。它只会创建匹配 `value` / `values` 长度所需的数量的手柄。例如，如果您指定 `values: [ 1, 5, 18 ]`，且创建一个自定义手柄，插件将创建其他两个。

主题化

滑块部件（Slider Widget）使用 jQuery UI CSS 框架 来定义它的外观和感观的样式。如果需要使用滑块指定的样式，则可以使用下面的 CSS class 名称：

- `ui-slider`：滑块控件的轨道。该元素会根据滑块的 `orientation` 另外带有一个 `ui-slider-horizontal` 或 `ui-slider-vertical` class。
  - `ui-slider-handle`：滑块手柄。
  - `ui-slider-range`：当设置 `range` 选项时使用的已选范围。如果 `range` 选项设置为 `"min"` 或 `"max"`，则该元素会分别另外带有一个 `ui-slider-range-min` 或 `ui-slider-range-max` class。

依赖

- UI 核心（UI Core）
- 部件库（Widget Factory）
- 鼠标交互（Mouse Interaction）

附加说明

- 该部件要求一些功能性的 CSS，否则将无法工作。如果您创建了一个自定义的主题，请使用小部件指定的 CSS 文件作为起点。

实例

一个简单的 jQuery UI 滑块（Slider）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>滑块部件（Slider Widget）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>#slider { margin: 10px; }    </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<div id="slider"></div>

<script>
$( "#slider" ).slider();
</script>

</body>
</html>
```

jQuery UI API - 旋转器部件（Spinner Widget）

所属类别

小部件（Widgets）

用法

描述：通过向上/向下按钮和箭头键处理，为输入数值增强文本输入功能。

版本新增：1.9

旋转器（Spinner），或数步进控件（number stepper widget），是用于处理各种数字输入的完美控件。它允许用户直接输入一个值，或通过键盘、鼠标、滚轮旋转改变一个已有的值。当与全球化（Globalize）结合时，您甚至可以旋转显示不同地区的货币和日期。

旋转器（Spinner）使用两个按钮将文本输入覆盖为当前值的递增值和递减值。旋转器增加了按键事件，以便可以用键盘完成相同的递增和递减。旋转器代表 全球化（Globalize）的数字格式和解析。

键盘交互

- UP：对值增加一步。
- DOWN：对值减少一步。
- PAGE UP：对值增加一页。
- PAGE DOWN：对值减少一页。

用鼠标点击旋转按钮后，焦点仍停留在文本域中。

当旋转器不是只读（ `<input readonly>` ）时，用户可以输入值，这可能会产生无效的值（小于最小值，大于最大值，增减错配，非数字输入）。当增减时，不管通过编程方式还是旋转按钮方式，值都会被强制为一个有效值（如需了解详情，请查看 `stepUp()` 和 `stepDown()` 的描述。

主题化

旋转器部件（Spinner Widget）使用 jQuery UI CSS 框架 来定义它的外观和感观的样式。如果需要使用旋转器指定的样式，则可以使用下面的 CSS class 名称：

- `ui-spinner` ：旋转器的外层容器。
  - `ui-spinner-input` ：旋转器部件（Spinner Widget）实例化的 `<input>` 元素。
  - `ui-spinner-button` ：用于递增或递减旋转器值的按钮控件。向上按钮

会另外带有一个 ui-spinner-up class，向下按钮会另外带有一个 ui-spinner-down class。

依赖

- UI 核心（UI Core）
- 部件库（Widget Factory）
- 按钮部件（Button Widget）
- 全球化（Globalize）（外部的，可选的；当与 culture 和 numberFormat 选项一起使用时）

附加说明

- 该部件要求一些功能性的 CSS，否则将无法工作。如果您创建了一个自定义的主题，请使用小部件指定的 CSS 文件作为起点。
- 该部件以编程方式操作元素的值，因此当元素的值改变时不会触发原生的 change 事件。
- 不支持在 <input type="number"> 上创建选择器，因为会造成与本地旋转器的 UI 冲突。

实例

普通的数字选择器。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>旋转器部件（Spinner Widget）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<input id="spinner">

<script>
$( "#spinner" ).spinner();
</script>

</body>
</html>
```

jQuery UI API - 标签页部件（Tabs Widget）

所属类别

小部件（Widgets）

用法

描述：一种多面板的单内容区，每个面板与列表中的标题相关。

版本新增：1.0

标签页（Tabs）通常用于把内容分成多个部分，以便节省空间，就像折叠面板（accordion）一样。

标签页（Tabs）有一组必须使用的特定标记，以便标签页能正常工作：

- 标签页（Tabs）必须在一个有序的（ `<ol>` ）或无序的（ `<ul>` ）列表中
- 每个标签页的 "title" 必须在一个列表项（ `<li>` ）的内部，且必须用一个带有 `href` 属性的锚（ `<a>` ）包裹。
- 每个标签页面板可以是任意有效的元素，但是它必须带有一个 id，该 id 与相关标签页的锚中的哈希相对应。

每个标签页面板的内容可以在页面中定义好，或者可以通过 Ajax 加载。这两种方式都是基于与标签页相关的锚的 `href` 上自动处理的。默认情况下，标签页在点击时激活，但是通过 `event` 选项可以改变或覆盖事件。

下面是一些样品标记：

```
<div id="tabs">
  <ul>
    <li><a href="#fragment-1">一</a></li>
    <li><a href="#fragment-2">二</a></li>
    <li><a href="#fragment-3">三</a></li>
  </ul>
  <div id="fragment-1">
    Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed d
  </div>
  <div id="fragment-2">
    Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed d
  </div>
  <div id="fragment-3">
    Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed d
  </div>
</div>
```

键盘交互

当焦点在标签页上时，下面的键盘命令可用：

- UP/LEFT：移动焦点到上一个标签页。如果在第一个标签页上，则移动焦点到最后一个标签页。在一个短暂的延迟后激活获得焦点的标签页。
- DOWN/RIGHT：移动焦点到下一个标签页。如果在最后一个标签页上，则移动焦点到第一个标签页。在一个短暂的延迟后激活获得焦点的标签页。
- HOME：移动焦点到第一个标签页。在一个短暂的延迟后激活获得焦点的标签页。
- END：移动焦点到最后一个标签页。在一个短暂的延迟后激活获得焦点的标签页。
- SPACE：激活与获得焦点的标签页相关的面板。
- ENTER：激活或切换与获得焦点的标签页相关的面板。
- ALT+PAGE UP：移动焦点到上一个标签页，并立即激活。
- ALT+PAGE DOWN：移动焦点到下一个标签页，并立即激活。

当焦点在面板上时，下面的键盘命令可用：

- CTRL+UP：移动焦点到相关的标签页。
- ALT+PAGE UP：移动焦点到上一个标签页，并立即激活。
- ALT+PAGE DOWN：移动焦点到下一个标签页，并立即激活。

主题化

标签页部件（Tabs Widget）使用 jQuery UI CSS 框架 来定义它的外观和感观的样式。如果需要使用标签页指定的样式，则可以使用下面的 CSS class 名称：

- `ui-tabs`：标签页的外层容器。当设置了 `collapsible` 选项时，该元素会另外带有一个 `ui-tabs-collapsible` class。

  - `ui-tabs-nav` ：标签页列表。
    - 导航中激活的列表项会带有一个 `ui-tabs-active` class。内容通过 Ajax 调用加载的列表项会带有一个 `ui-tabs-loading` class。
      - `ui-tabs-anchor` ：用于切换面板的锚。
  - `ui-tabs-panel` ：与标签页相关的面板。只有与其对应的标签页激活时才可见。

依赖

- UI 核心（UI Core）
- 部件库（Widget Factory）
- 特效核心（Effects Core）（可选的；当与 `show` 和 `hide` 选项一起使用时）

附加说明

- 该部件要求一些功能性的 CSS，否则将无法工作。如果您创建了一个自定义的主题，请使用小部件指定的 CSS 文件作为起点。

实例

一个简单的 jQuery UI 标签页（Tabs）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>标签页部件（Tabs Widget）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<div id="tabs">
  <ul>
    <li><a href="#fragment-1"><span>一</span></a></li>
    <li><a href="#fragment-2"><span>二</span></a></li>
    <li><a href="#fragment-3"><span>三</span></a></li>
  </ul>
  <div id="fragment-1">
    <p>第一个标签是默认激活的：</p>
    <code>$( "#tabs" ).tabs(); </code>
  </div>
  <div id="fragment-2">
    Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed d
    Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed d
  </div>
  <div id="fragment-3">
    Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed d
    Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed d
    Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed d
  </div>
</div>

<script>
$( "#tabs" ).tabs();
</script>

</body>
</html>
```

jQuery UI API - 工具提示框部件（Tooltip Widget）

所属类别

小部件（Widgets）

用法

描述：可自定义的、可主题化的工具提示框，替代原生的工具提示框。

版本新增：1.9

工具提示框（Tooltip）取代了原生的工具提示框，让它们可主题化，也允许进行各种自定义：

- 显示不仅仅是标题的其他内容，就如内联的脚注或通过 Ajax 检索的额外内容。
- 自定义定位，例如，在元素上居中工具提示框。
- 添加额外的样式来定制警告或错误区域的外观。

默认使用一个渐变的动画来显示和隐藏工具提示框，这种外观与简单的切换可见度相比更具灵性。这可以通过 `show` 和 `hide` 选项进行定制。

`items` 和 `content` 选项需要保持同步。如果您改变了其中一个，您需要同时改变另一个。

在一般情况下，禁用的元素不会触发任何 DOM 事件。因此，适当地控制禁用元素的工具提示框是不可能的，因为我们需要监听事件来决定何时显示和隐藏工具提示框。这就导致 jQuery UI 不能保证对附加到禁用元素上的工具提示框任何层次上的支持。这意味着如果您需要在禁用元素上进行提示，您可能需要使用一个原生的提示框和 jQuery UI 工具提示框的混合物。

主题化

工具提示框部件（Tooltip Widget）使用 jQuery UI CSS 框架 来定义它的外观和感观的样式。如果需要使用工具提示框指定的样式，则可以使用下面的 CSS class 名称：

- `ui-tooltip` ：工具提示框的外层容器。
    - `ui-tooltip-content` ：工具提示框的内容。

依赖

- UI 核心（UI Core）
- 部件库（Widget Factory）
- 定位（Position）
- 特效核心（Effects Core）（可选的；当与 `show` 和 `hide` 选项一起使用时）

附加说明

- 该部件要求一些功能性的 CSS，否则将无法工作。如果您创建了一个自定义的主题，请使用小部件指定的 CSS 文件作为起点。

实例

使用带有 title 属性的所有元素的事件代理，在文档上创建一个工具提示框（Tooltip）。

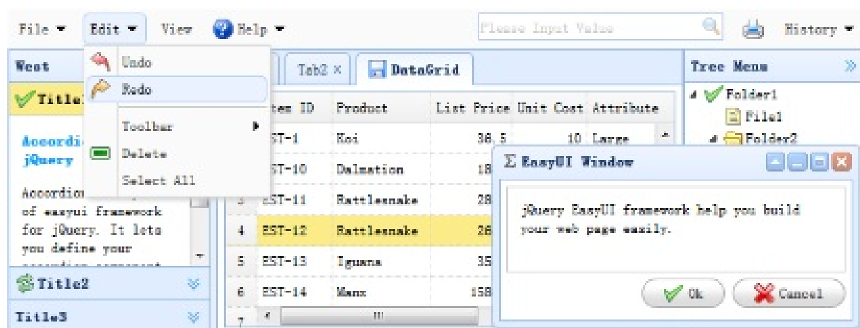
```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>工具提示框部件（Tooltip Widget）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<p>
  <a href="#" title="锚描述">锚文本</a>
  <input title="输入帮助">
</p>
<script>
  $( document ).tooltip();
</script>

</body>
</html>
```

jQuery EasyUI 简介

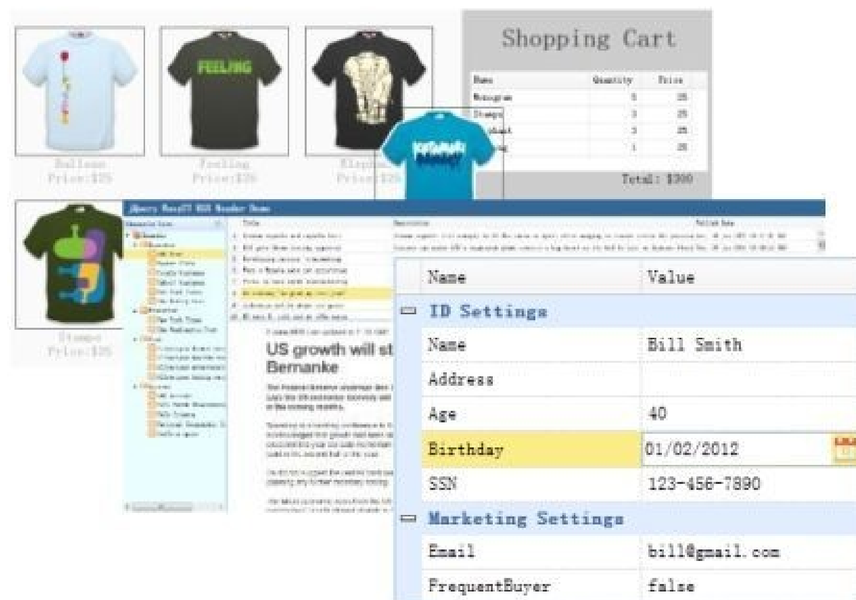
jQuery EasyUI 是一个基于 jQuery 的框架，集成了各种用户界面插件。



什么是 jQuery EasyUI

jQuery EasyUI 框架提供了创建网页所需的一切，帮助您轻松建立站点。

- easyui 是一个基于 jQuery 的框架，集成了各种用户界面插件。
- easyui 提供建立现代化的具有交互性的 javascript 应用的必要的功能。
- 使用 easyui，您不需要写太多 javascript 代码，一般情况下您只需要使用一些 html 标记来定义用户界面。
- HTML 网页的完整框架。
- easyui 节省了开发产品的时间和规模。
- easyui 非常简单，但是功能非常强大。



jQuery EasyUI 下载

您可以从 http://www.jeasyui.com/download/index.php 上下载您需要的 jQuery EasyUI 版本。

轻松使用 jQuery 和 HTML5

jQuery EasyUI 提供易于使用的组件，它使 Web 开发人员快速地在流行的 jQuery 核心和 HTML5 上建立程序页面。 这些功能使您的应用适合今天的网络。 有两个方法声明的 UI 组件:

1. 直接在 HTML 声明组件。

```
<div class="easyui-dialog" style="width:400px;height:200px"
    data-options="title:'My Dialog',collapsible:true,iconCls:'icon-
        dialog content.
</div>
```

2. 编写 JavaScript 代码来创建组件。

```
<input id="cc" style="width:200px" />
```

```
$('#cc').combobox({
    url: ...,
    required: true,
    valueField: 'id',
    textField: 'text'
});
```

jQuery EasyUI 应用

jQuery EasyUI 应用 - 创建 CRUD 应用

数据收集并妥善管理数据是网络应用共同的必要。CRUD 允许我们生成页面列表，并编辑数据库记录。本教程将向你演示如何使用 jQuery EasyUI 框架实现一个 CRUD DataGrid。

我们将使用下面的插件：

- datagrid：向用户展示列表数据。
- dialog：创建或编辑一条单一的用户信息。
- form：用于提交表单数据。
- messager：显示一些操作信息。

步骤 1：准备数据库

我们将使用 MySql 数据库来存储用户信息。创建数据库和 'users' 表。

id	firstname	lastname	phone	email
3	fname1	lname1	(000)000-0000	name1@gmail.com
4	fname2	lname2	(000)000-0000	name2@gmail.com
5	fname3	lname3	(000)000-0000	name3@gmail.com
7	fname4	lname4	(000)000-0000	name4@gmail.com
8	fname5	lname5	(000)000-0000	name5@gmail.com
9	fname6	lname6	(000)000-0000	name6@gmail.com
10	fname7	lname7	(000)000-0000	name7@gmail.com
11	fname8	lname8	(000)000-0000	name8@gmail.com
12	fname9	lname9	(000)000-0000	name9@gmail.com
13	fname10	lname10	(000)000-0000	name10@gmail.com

步骤 2：创建 DataGrid 来显示用户信息

创建没有 javascript 代码的 DataGrid。

```
<table id="dg" title="My Users" class="easyui-datagrid" style="widt
        url="get_users.php"
        toolbar="#toolbar"
        rownumbers="true" fitColumns="true" singleSelect="true">
    <thead>
        <tr>
            <th field="firstname" width="50">First Name</th>
            <th field="lastname" width="50">Last Name</th>
            <th field="phone" width="50">Phone</th>
            <th field="email" width="50">Email</th>
        </tr>
    </thead>
</table>
<div id="toolbar">
    <a href="#" class="easyui-linkbutton" iconCls="icon-add" plain=
    <a href="#" class="easyui-linkbutton" iconCls="icon-edit" plain
    <a href="#" class="easyui-linkbutton" iconCls="icon-remove" pla
</div>
```

我们不需要写任何的 javascript 代码，就能向用户显示列表，如下图所示：

My Users

New User　Edit User　Remove User

	First Name	Last Name	Phone	Email
1	fname1	lname1	(000)000-0000	name1@gmail.com
2	fname2	lname2	(000)000-0000	name2@gmail.com
3	fname3	lname3	(000)000-0000	name3@gmail.com
4	fname4	lname4	(000)000-0000	name4@gmail.com
5	fname5	lname5	(000)000-0000	name5@gmail.com
6	fname6	lname6	(000)000-0000	name6@gmail.com
7	fname7	lname7	(000)000-0000	name7@gmail.com
8	fname8	lname8	(000)000-0000	name8@gmail.com

DataGrid 使用 'url' 属性，并赋值为 'get_users.php'，用来从服务器检索数据。

get_users.php 文件的代码

```
$rs = mysql_query('select * from users');
$result = array();
while($row = mysql_fetch_object($rs)){
    array_push($result, $row);
}

echo json_encode($result);
```

步骤 3：创建表单对话框

我们使用相同的对话框来创建或编辑用户。

```
<div id="dlg" class="easyui-dialog" style="width:400px;height:280px
        closed="true" buttons="#dlg-buttons">
    <div class="ftitle">User Information</div>
    <form id="fm" method="post">
        <div class="fitem">
            <label>First Name:</label>
            <input name="firstname" class="easyui-validatebox" requ
        </div>
        <div class="fitem">
            <label>Last Name:</label>
            <input name="lastname" class="easyui-validatebox" requi
        </div>
        <div class="fitem">
            <label>Phone:</label>
            <input name="phone">
        </div>
        <div class="fitem">
            <label>Email:</label>
            <input name="email" class="easyui-validatebox" validTyp
        </div>
    </form>
</div>
<div id="dlg-buttons">
    <a href="#" class="easyui-linkbutton" iconCls="icon-ok" onclick
    <a href="#" class="easyui-linkbutton" iconCls="icon-cancel" onc
</div>
```

这个对话框已经创建，也没有任何的 javascript 代码:。

步骤 4：实现创建和编辑用户

当创建用户时，打开一个对话框并清空表单数据。

```
function newUser(){
    $('#dlg').dialog('open').dialog('setTitle','New User');
    $('#fm').form('clear');
    url = 'save_user.php';
}
```

当编辑用户时，打开一个对话框并从 datagrid 选择的行中加载表单数据。

```
var row = $('#dg').datagrid('getSelected');
if (row){
    $('#dlg').dialog('open').dialog('setTitle','Edit User');
    $('#fm').form('load',row);
    url = 'update_user.php?id='+row.id;
}
```

'url' 存储着当保存用户数据时表单回传的 URL 地址。

步骤 5：保存用户数据

我们使用下面的代码保存用户数据：

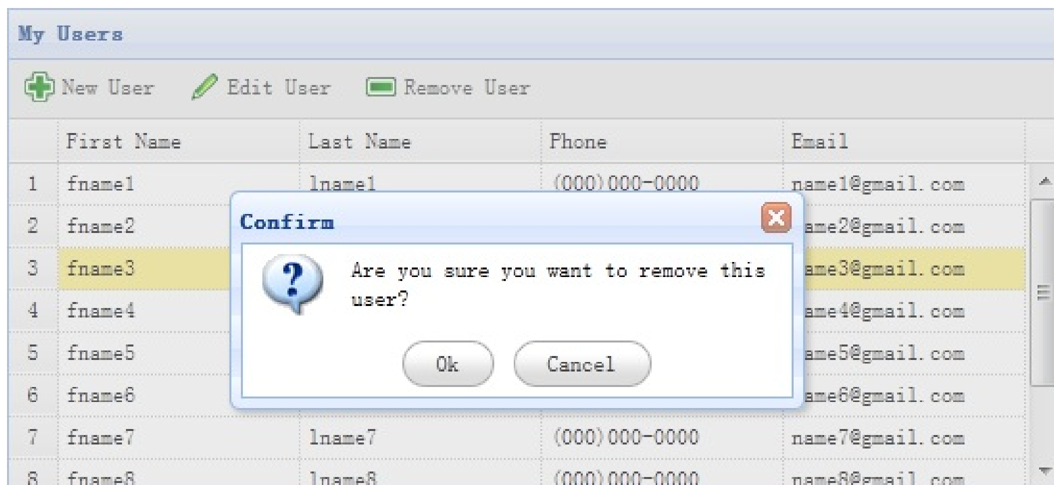
```
function saveUser(){
    $('#fm').form('submit',{
        url: url,
        onSubmit: function(){
            return $(this).form('validate');
        },
        success: function(result){
            var result = eval('('+result+')');
            if (result.errorMsg){
                $.messager.show({
                    title: 'Error',
                    msg: result.errorMsg
                });
            } else {
                $('#dlg').dialog('close');        // close the dial
                $('#dg').datagrid('reload');    // reload the user
            }
        }
    });
}
```

提交表单之前，'onSubmit' 函数将被调用，该函数用来验证表单字段值。当表单字段值提交成功，关闭对话框并重新加载 datagrid 数据。

步骤 6：删除一个用户

我们使用下面的代码来移除一个用户：

```
function destroyUser(){
    var row = $('#dg').datagrid('getSelected');
    if (row){
        $.messager.confirm('Confirm','Are you sure you want to dest
            if (r){
                $.post('destroy_user.php',{id:row.id},function(resu
                    if (result.success){
                        $('#dg').datagrid('reload');    // reload t
                    } else {
                        $.messager.show({    // show error message
                            title: 'Error',
                            msg: result.errorMsg
                        });
                    }
                },'json');
            }
        });
    }
}
```



移除一行之前,我们将显示一个确认对话框让用户决定是否真的移除该行数据。当移除数据成功之后，调用 'reload' 方法来刷新 datagrid 数据。

步骤 7：运行代码

开启 MySQL，在浏览器运行代码。

下载 jQuery EasyUI 实例

jeasyui-app-crud1.zip

jQuery EasyUI 应用 - 创建 CRUD 数据网格（DataGrid）

在上一章节中，我们使用对话框（dialog）组件创建了 CRUD 应用来创建和编辑用户信息。本教程我们将告诉您如何创建一个 CRUD 数据网格（DataGrid）。 我们将使用 可编辑的数据网格（DataGrid）插件 来完成这些 CRUD 操作动作。

My Users

New　Destroy　Save　Cancel

	First Name	Last Name	Phone	Email
3	fname3	lname3	(000)000-0000	name3@gmail.com
4	fname4	lname4	(000)000-0000	name4@gmail.com
5	fname5	This field is required.		@gmail.com
6	fname6	lname6	(000)000-0000	name6@gmail.com
7	fname7	lname7	(000)000-0000	name7@gmail.com
8	fname8	lname8	(000)000-0000	name8@gmail.com
9	fname9	lname9	(000)000-0000	name9@gmail.com
10	fname10	lname10	(000)000-0000	name10@gmail.com

步骤 1：在 HTML 标签中定义数据网格（DataGrid）

```
<table id="dg" title="My Users" style="width:550px;height:250px"
        toolbar="#toolbar" idField="id"
        rownumbers="true" fitColumns="true" singleSelect="true">
    <thead>
        <tr>
            <th field="firstname" width="50" editor="{type:'validat
            <th field="lastname" width="50" editor="{type:'validate
            <th field="phone" width="50" editor="text">Phone</th>
            <th field="email" width="50" editor="{type:'validatebox
        </tr>
    </thead>
</table>
<div id="toolbar">
    <a href="#" class="easyui-linkbutton" iconCls="icon-add" plain=
    <a href="#" class="easyui-linkbutton" iconCls="icon-remove" pla
    <a href="#" class="easyui-linkbutton" iconCls="icon-save" plain
    <a href="#" class="easyui-linkbutton" iconCls="icon-undo" plain
</div>
```

步骤 2：使用可编辑的数据网格（DataGrid）

```
$('#dg').edatagrid({
    url: 'get_users.php',
    saveUrl: 'save_user.php',
    updateUrl: 'update_user.php',
    destroyUrl: 'destroy_user.php'
});
```

我们应该提供 'url'、'saveUrl'、'updateUrl' 和 'destroyUrl' 属性来编辑数据网格（DataGrid）：

- url：从服务器端检索用户数据。
- saveUrl：保存一个新的用户数据。
- updateUrl：更新一个已存在的用户数据。
- destroyUrl：删除一个已存在的用户数据。

步骤 3：写服务器处理代码

保存一个新的用户（save_user.php）：

```
$firstname = $_REQUEST['firstname'];
$lastname = $_REQUEST['lastname'];
$phone = $_REQUEST['phone'];
$email = $_REQUEST['email'];

include 'conn.php';

$sql = "insert into users(firstname,lastname,phone,email) values('$
@mysql_query($sql);
echo json_encode(array(
    'id' => mysql_insert_id(),
    'firstname' => $firstname,
    'lastname' => $lastname,
    'phone' => $phone,
    'email' => $email
));
```

更新一个已存在用户（update_user.php）：

```
$id = intval($_REQUEST['id']);
$firstname = $_REQUEST['firstname'];
$lastname = $_REQUEST['lastname'];
$phone = $_REQUEST['phone'];
$email = $_REQUEST['email'];

include 'conn.php';

$sql="update users set firstname='$firstname',lastname='$lastname',
@mysql_query($sql);
echo json_encode(array(
    'id' => $id,
    'firstname' => $firstname,
    'lastname' => $lastname,
    'phone' => $phone,
    'email' => $email
));
```

删除一个已存在用户（destroy_user.php）：

```
$id = intval($_REQUEST['id']);

include 'conn.php';

$sql = "delete from users where id=$id";
@mysql_query($sql);
echo json_encode(array('success'=>true));
```

下载 jQuery EasyUI 实例

jeasyui-app-crud2.zip

jQuery EasyUI 应用 - 创建展开行明细编辑表单的 CRUD 应用

当切换数据网格视图（datagrid view）到 'detailview'，用户可以展开一行来显示一些行的明细在行下面。这个功能允许您为防止在明细行面板（panel）中的编辑表单（form）提供一些合适的布局（layout）。在本教程中，我们使用数据网格（datagrid）组件来减小编辑表单（form）所占据空间。

步骤 1：在 HTML 标签中定义数据网格（DataGrid）

```
<table id="dg" title="My Users" style="width:550px;height:250px"
        url="get_users.php"
        toolbar="#toolbar"
        fitColumns="true" singleSelect="true">
    <thead>
        <tr>
            <th field="firstname" width="50">First Name</th>
            <th field="lastname" width="50">Last Name</th>
            <th field="phone" width="50">Phone</th>
            <th field="email" width="50">Email</th>
        </tr>
    </thead>
</table>
<div id="toolbar">
    <a href="#" class="easyui-linkbutton" iconCls="icon-add" plain=
    <a href="#" class="easyui-linkbutton" iconCls="icon-remove" pla
</div>
```

步骤 2：为数据网格（DataGrid）应用明细视图

```
$('#dg').datagrid({
    view: detailview,
    detailFormatter:function(index,row){
        return '<div class="ddv"></div>';
    },
    onExpandRow: function(index,row){
        var ddv = $(this).datagrid('getRowDetail',index).find('div
        ddv.panel({
            border:false,
            cache:true,
            href:'show_form.php?index='+index,
            onLoad:function(){
                $('#dg').datagrid('fixDetailRowHeight',index);
                $('#dg').datagrid('selectRow',index);
                $('#dg').datagrid('getRowDetail',index).find('form
            }
        });
        $('#dg').datagrid('fixDetailRowHeight',index);
    }
});
```

为了为数据网格（DataGrid）应用明细视图，在 html 页面头部引入 'datagrid-detailview.js' 文件。

我们使用 'detailFormatter' 函数来生成行明细内容。 在这种情况下，我们返回一个用于放置编辑表单（form）的空的 <div>。 当用户点击行展开按钮（'+'）时，'onExpandRow' 事件将被触发，我们将通过 ajax 加载编辑表单（form）。 调用 'getRowDetail' 方法来得到行明细容器，所以我们能查找到行明细面板（panel）。 在行明细中创建面板（panel），加载从 'show_form.php' 返回的编辑表单（form）。

步骤 3：创建编辑表单（Form）

编辑表单（form）是从服务器加载的。

show_form.php

```
<form method="post">
    <table class="dv-table" style="width:100%;background:#fafafa;pa
        <tr>
            <td>First Name</td>
            <td><input name="firstname" class="easyui-validatebox"
            <td>Last Name</td>
            <td><input name="lastname" class="easyui-validatebox" r
        </tr>
        <tr>
            <td>Phone</td>
            <td><input name="phone"></input></td>
            <td>Email</td>
            <td><input name="email" class="easyui-validatebox" vali
        </tr>
    </table>
    <div style="padding:5px 0;text-align:right;padding-right:30px">
        <a href="#" class="easyui-linkbutton" iconCls="icon-save" p
        <a href="#" class="easyui-linkbutton" iconCls="icon-cancel"
    </div>
</form>
```

步骤 4：保存或取消编辑

调用 'saveItem' 函数来保存一个用户或者调用 'cancelItem' 函数来取消编辑。

```
function saveItem(index){
    var row = $('#dg').datagrid('getRows')[index];
    var url = row.isNewRecord ? 'save_user.php' : 'update_user.php?
    $('#dg').datagrid('getRowDetail',index).find('form').form('subm
        url: url,
        onSubmit: function(){
            return $(this).form('validate');
        },
        success: function(data){
            data = eval('('+data+')');
            data.isNewRecord = false;
            $('#dg').datagrid('collapseRow',index);
            $('#dg').datagrid('updateRow',{
                index: index,
                row: data
            });
        }
    });
}
```

决定要回传哪一个 URL，然后查找表单（form）对象，并调用 'submit' 方法来提交表单（form）数据。当保存数据成功时，折叠并更新行数据。

```
function cancelItem(index){
    var row = $('#dg').datagrid('getRows')[index];
    if (row.isNewRecord){
        $('#dg').datagrid('deleteRow',index);
    } else {
        $('#dg').datagrid('collapseRow',index);
    }
}
```

当取消编辑动作时，如果该行是新行而且还没有保存，直接删除该行，否则折叠该行。

下载 jQuery EasyUI 实例

jeasyui-app-crud3.zip

jQuery EasyUI 应用 - 创建 RSS Feed 阅读器

在本教程中，我们将通过 jQuery EasyUI 框架创建一个 RSS 阅读器。



我们将使用以下插件:

- layout：创建应用的用户界面。
- datagrid：显示 RSS Feed 列表。
- tree：显示 feed 频道。

步骤 1：创建布局（Layout）

```
&lt;body class="easyui-layout"&gt;
    <div region="north" border="false" class="rtitle">
        jQuery EasyUI RSS Reader Demo
    </div>
    <div region="west" title="Channels Tree" split="true" border="f
        <ul id="t-channels" url="data/channels.json"></ul>
    </div>
    <div region="center" border="false">
        <div class="easyui-layout" fit="true">
            <div region="north" split="true" border="false" style='
                <table id="dg"
                        url="get_feed.php" border="false" rownumber
                        fit="true" fitColumns="true" singleSelect='
                    <thead>
                        <tr>
                            <th field="title" width="100">Title</t
                            <th field="description" width="200">Des
                            <th field="pubdate" width="80">Publish
                        </tr>
                    </thead>
                </table>
            </div>
            <div region="center" border="false" style="overflow:hid
                <iframe id="cc" scrolling="auto" frameborder="0" st
            </div>
        </div>
    </div>
&lt;/body&gt;
```

步骤 2：数据网格（DataGrid）处理事件

在这里我们要处理一些由用户触发的事件。

```
$('#dg').datagrid({
    onSelect: function(index,row){
        $('#cc').attr('src', row.link);
    },
    onLoadSuccess:function(){
        var rows = $(this).datagrid('getRows');
        if (rows.length){
            $(this).datagrid('selectRow',0);
        }
    }
});
```

本实例使用 'onSelect' 事件来显示 feed 的内容，使用 'onLoadSuccess' 事件来选择第一行。

步骤 3：树形菜单（Tree）处理事件

当树形菜单（Tree）数据已经加载，我们需要选择第一个叶子节点，调用 'select' 方法来选择该节点。 使用 'onSelect' 事件来得到已选择的节点，这样我们就能得到对应的 'url' 值。 最后我们调用数据网格（DataGrid） 的 'load' 方法来刷新 feed 列表数据。

```
$('#t-channels').tree({
    onSelect: function(node){
        var url = node.attributes.url;
        $('#dg').datagrid('load',{
            url: url
        });
    },
    onLoadSuccess:function(node,data){
        if (data.length){
            var id = data[0].children[0].children[0].id;
            var n = $(this).tree('find', id);
            $(this).tree('select', n.target);
        }
    }
});
```

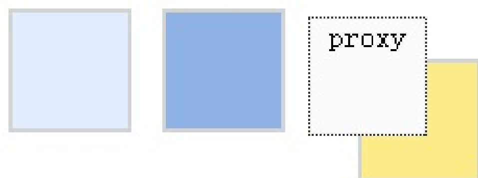
下载 jQuery EasyUI 实例

jeasyui-app-rssreader.zip

jQuery EasyUI 拖放

jQuery EasyUI 拖放 - 基本的拖动和放置

本教程向您展示如何使 HTML 元素可拖动，在本例中，我们将创建三个 DIV 元素然后启用他们的拖动和放置。



首先，我们创建三个 <div> 元素：

```
<div id="dd1" class="dd-demo"></div>
<div id="dd2" class="dd-demo"></div>
<div id="dd3" class="dd-demo"></div>
```

对于第一个 <div> 元素，我们通过默认值让其可以拖动。

```
$('#dd1').draggable();
```

对于第二个 <div> 元素，我们通过创建一个克隆（clone）了原来元素的代理（proxy）让其可以拖动。

```
$('#dd2').draggable({
    proxy:'clone'
});
```

对于第三个 <div> 元素，我们通过创建自定义代理（proxy）让其可以拖动。

```
$('#dd3').draggable({
    proxy:function(source){
        var p = $('<div class="proxy">proxy</div>');
        p.appendTo('body');
        return p;
    }
});
```

下载 jQuery EasyUI 实例

jeasyui-dd-basic.zip

jQuery EasyUI 拖放 - 创建拖放的购物车

如果您能够通过您的 Web 应用简单地实现拖动和放置，您就会知道一些特别的东西。通过 jQuery EasyUI，我们在 Web 应用中可以简单地实现拖放功能。

在本教程中，我们将向您展示如何创建一个启用用户拖动和放置用户想买的商品的购物车页面。购物篮中的物品和价格将更新。



显示页面上的商品

```
<ul class="products">
    <li>
        <a href="#" class="item">
            <img src="images/shirt1.gif"/>
            <div>
                <p>Balloon</p>
                <p>Price:$25</p>
            </div>
        </a>
    </li>
    <li>
        <a href="#" class="item">
            <img src="images/shirt2.gif"/>
            <div>
                <p>Feeling</p>
                <p>Price:$25</p>
            </div>
        </a>
    </li>
    <!-- other products -->
</ul>
```

正如您所看到的上面的代码，我们添加一个包含一些 <li> 元素的 <ul> 元素来显示商品。所有商品都有名字和价格属性，它们包含在 <p> 元素中。

创建购物车

```
<div class="cart">
    <h1>Shopping Cart</h1>
    <table id="cartcontent" style="width:300px;height:auto;">
        <thead>
            <tr>
                <th field="name" width=140>Name</th>
                <th field="quantity" width=60 align="right">Qua
                <th field="price" width=60 align="right">Price<
            </tr>
        </thead>
    </table>
    <p class="total">Total: $0</p>
    <h2>Drop here to add to cart</h2>
</div>
```

我们使用数据网格（datagrid）来显示购物篮中的物品。

拖动克隆的商品

```
$('.item').draggable({
    revert:true,
    proxy:'clone',
    onStartDrag:function(){
        $(this).draggable('options').cursor = 'not-allowed';
        $(this).draggable('proxy').css('z-index',10);
    },
    onStopDrag:function(){
        $(this).draggable('options').cursor='move';
    }
});
```

请注意，我们把 draggable 属性的值从 'proxy' 设置为 'clone'，所以拖动元素将由克隆产生。

放置选择商品到购物车中

```
    $('.cart').droppable({
        onDragEnter:function(e,source){
            $(source).draggable('options').cursor='auto';
        },
        onDragLeave:function(e,source){
            $(source).draggable('options').cursor='not-allowed';
        },
        onDrop:function(e,source){
            var name = $(source).find('p:eq(0)').html();
            var price = $(source).find('p:eq(1)').html();
            addProduct(name, parseFloat(price.split('$')[1]));
        }
    });
    var data = {"total":0,"rows":[]};
    var totalCost = 0;
    function addProduct(name,price){
        function add(){
            for(var i=0; i<data.total; i++){
                var row = data.rows[i];
                if (row.name == name){
                    row.quantity += 1;
                    return;
                }
            }
            data.total += 1;
            data.rows.push({
                name:name,
                quantity:1,
                price:price
            });
        }
        add();
        totalCost += price;
        $('#cartcontent').datagrid('loadData', data);
        $('div.cart .total').html('Total: $'+totalCost);
    }
```

每当放置商品的时候，我们首先得到商品名称和价格，然后调用 'addProduct' 函数来更新购物篮。

下载 jQuery EasyUI 实例

jeasyui-dd-shopping.zip

jQuery EasyUI 拖放 - 创建学校课程表

本教程将向您展示如何使用 jQuery EasyUI 创建一个学校课程表。 我们将创建两个表格：在左侧显示学校科目，在右侧显示时间表。 您可以拖动学校科目并放置到时间表单元格上。 学校科目是一个 <div class="item"> 元素，时间表单元格是一个 <td class="drop"> 元素。

English
Science
Music
History
Computer
Mathematics
Arts
Ethics

	Monday	Tuesday	Wednesday	Thursday	Friday
08:00					
09:00	Science				
10:00	English			Mathematics	
11:00		History			
12:00	Computer				
13:00			Lunch		
14:00					
15:00			Arts		
16:00					

显示学校科目

```
<div class="left">
    <table>
        <tr>
            <td><div class="item">English</div></td>
        </tr>
        <tr>
            <td><div class="item">Science</div></td>
        </tr>
        <!-- other subjects -->
    </table>
</div>
```

显示时间表

```
    <div class="right">
        <table>
            <tr>
                <td class="blank"></td>
                <td class="title">Monday</td>
                <td class="title">Tuesday</td>
                <td class="title">Wednesday</td>
                <td class="title">Thursday</td>
                <td class="title">Friday</td>
            </tr>
            <tr>
                <td class="time">08:00</td>
                <td class="drop"></td>
                <td class="drop"></td>
                <td class="drop"></td>
                <td class="drop"></td>
                <td class="drop"></td>
            </tr>
            <!-- other cells -->
        </table>
    </div>
```

拖动在左侧的学校科目

```
    $('.left .item').draggable({
        revert:true,
        proxy:'clone'
    });
```

放置学校科目在时间表单元格上

```
$('.right td.drop').droppable({
    onDragEnter:function(){
        $(this).addClass('over');
    },
    onDragLeave:function(){
        $(this).removeClass('over');
    },
    onDrop:function(e,source){
        $(this).removeClass('over');
        if ($(source).hasClass('assigned')){
            $(this).append(source);
        } else {
            var c = $(source).clone().addClass('assigned');
            $(this).empty().append(c);
            c.draggable({
                revert:true
            });
        }
    }
});
```

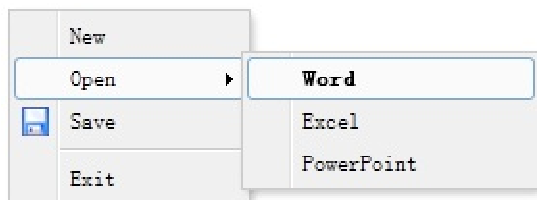
正如您所看到的上面的代码，当用户拖动在左侧的学校科目并放置到时间表单元格中时，onDrop 回调函数将被调用。我们克隆从左侧拖动的源元素并把它附加到时间表单元格上。 当把学校科目从时间表的某个单元格拖动到其他单元格，只需简单地移动它即可。

下载 jQuery EasyUI 实例

jeasyui-dd-timetable.zip

jQuery EasyUI 菜单与按钮

jQuery EasyUI 菜单与按钮 - 创建简单的菜单



菜单（Menu）定义在一些 DIV 标记中，如下所示：

```
<div id="mm" class="easyui-menu" style="width:120px;">
    <div onclick="javascript:alert('new')">New</div>
    <div>
        <span>Open</span>
        <div style="width:150px;">
            <div><b>Word</b></div>
            <div>Excel</div>
            <div>PowerPoint</div>
        </div>
    </div>
    <div icon="icon-save">Save</div>
    <div class="menu-sep"></div>
    <div>Exit</div>
</div>
```

当菜单创建之后是不显示的，调用 'show' 方法显示它或者调用 'hide' 方法隐藏它：

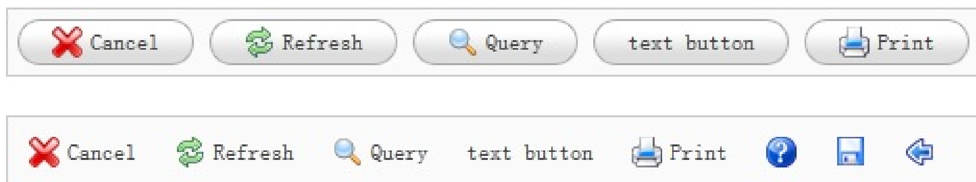
```
$('#mm').menu('show', {
  left: 200,
  top: 100
});
```

下载 jQuery EasyUI 实例

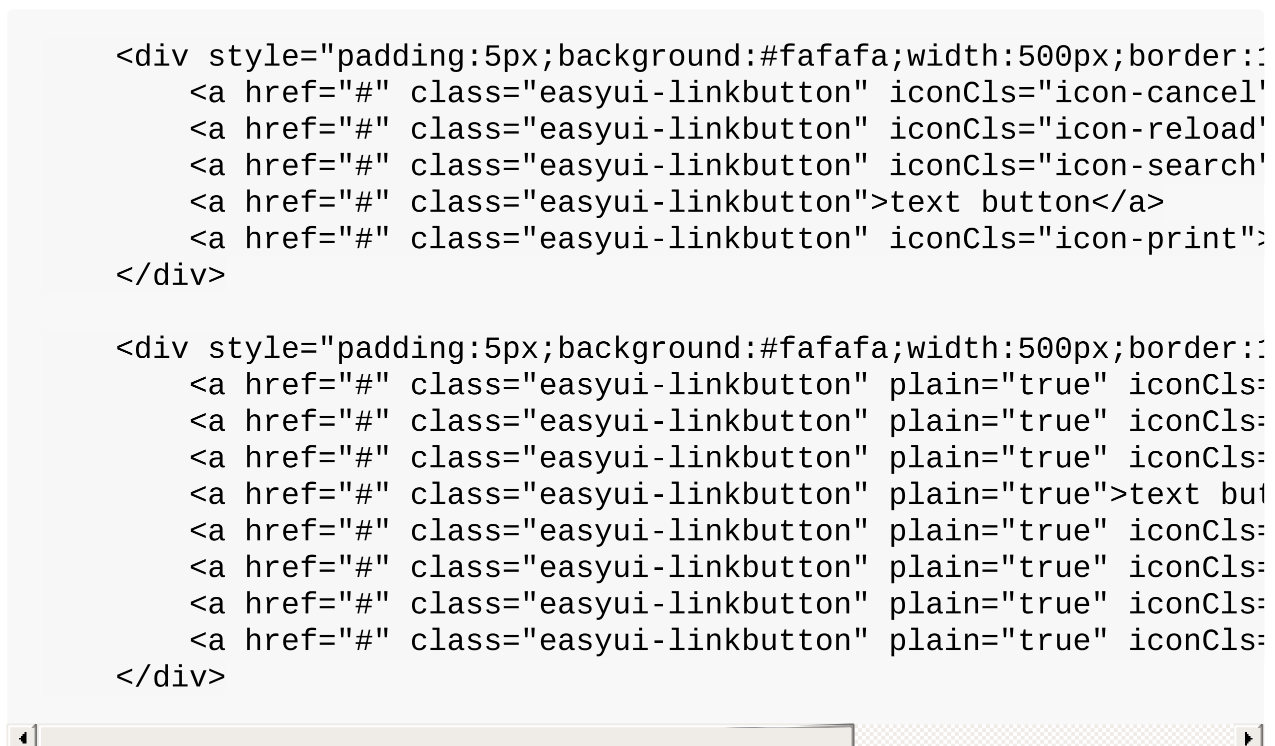
jeasyui-mb-menu.zip

jQuery EasyUI 菜单与按钮 - 创建链接按钮（Link Button）

通常情况下，使用 <button> 元素来创建按钮，而链接按钮（Link Button）则是使用 <a> 元素来创建的。所以实际上一个链接按钮（Link Button）就是一个显示为按钮样式的 <a> 元素。



为了创建链接按钮（Link Button），所有您需要做的就是添加一个名为 'easyui-linkbutton' 的 class 属性到 <a> 元素：



```
<div style="padding:5px;background:#fafafa;width:500px;border:1
    <a href="#" class="easyui-linkbutton" iconCls="icon-cancel'
    <a href="#" class="easyui-linkbutton" iconCls="icon-reload'
    <a href="#" class="easyui-linkbutton" iconCls="icon-search'
    <a href="#" class="easyui-linkbutton">text button</a>
    <a href="#" class="easyui-linkbutton" iconCls="icon-print">
</div>

<div style="padding:5px;background:#fafafa;width:500px;border:1
    <a href="#" class="easyui-linkbutton" plain="true" iconCls=
    <a href="#" class="easyui-linkbutton" plain="true" iconCls=
    <a href="#" class="easyui-linkbutton" plain="true" iconCls=
    <a href="#" class="easyui-linkbutton" plain="true">text but
    <a href="#" class="easyui-linkbutton" plain="true" iconCls=
    <a href="#" class="easyui-linkbutton" plain="true" iconCls=
    <a href="#" class="easyui-linkbutton" plain="true" iconCls=
    <a href="#" class="easyui-linkbutton" plain="true" iconCls=
</div>
```

正如您所看到的，iconCls 属性是一个 icon 的 CSS class 样式，它在按钮上显示一个 icon 图片。

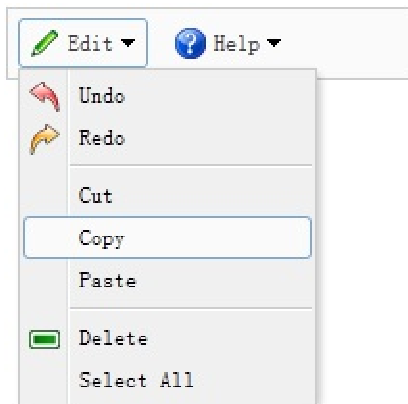
有时候您需要禁用链接按钮（Link Button）或者启用它，下面的代码演示了如何禁用一个链接按钮（Link Button）：



```
$(selector).linkbutton('disable');    // call the 'disable' met
```

jQuery EasyUI 菜单与按钮 - 创建菜单按钮（Menu Button）

菜单按钮（Menu Button）包含一个按钮（button）和一个菜单（menu）组件，当点击或移动鼠标到按钮上，将显示一个对应的菜单。



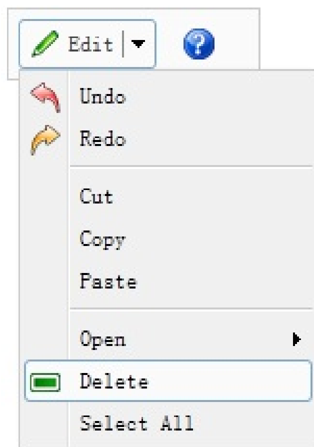
为了定义一个菜单按钮（Menu Button），您应该定义一个链接按钮（Link Button）和一个菜单（menu），下面是一个实例：

```
<div style="background:#fafafa;padding:5px;width:200px;border:1
    <a href="#" class="easyui-menubutton" menu="#mm1" iconCls='
    <a href="#" class="easyui-menubutton" menu="#mm2" iconCls='
</div>
<div id="mm1" style="width:150px;">
    <div iconCls="icon-undo">Undo</div>
    <div iconCls="icon-redo">Redo</div>
    <div class="menu-sep"></div>
    <div>Cut</div>
    <div>Copy</div>
    <div>Paste</div>
    <div class="menu-sep"></div>
    <div iconCls="icon-remove">Delete</div>
    <div>Select All</div>
</div>
<div id="mm2" style="width:100px;">
    <div>Help</div>
    <div>Update</div>
    <div>About</div>
</div>
```

现在已经定义好了一个菜单按钮（Menu Button），您不需要写任何的 javascript 代码。

jQuery EasyUI 菜单与按钮 - 创建分割按钮（Split Button）

分割按钮（Split Button）包含一个链接按钮（Link Button）和一个菜单（Menu）。当用户点击或者鼠标悬停在向下箭头区域，将会显示一个对应的菜单。本实例演示了如何创建和使用分割按钮（Split Button）。



我们创建一个分割按钮（Split Button）和一个链接按钮（Link Button）：

```
    <div style="border:1px solid #ccc;background:#fafafa;padding:5p
        <a href="#" class="easyui-splitbutton" menu="#mm" iconCls='
        <a href="#" class="easyui-linkbutton" plain="true" iconCls=
    </div>
    <div id="mm" style="width:150px;">
        <div iconCls="icon-undo">Undo</div>
        <div iconCls="icon-redo">Redo</div>
        <div class="menu-sep"></div>
        <div>Cut</div>
        <div>Copy</div>
        <div>Paste</div>
        <div class="menu-sep"></div>
        <div>
            <span>Open</span>
            <div style="width:150px;">
                <div>Firefox</div>
                <div>Internet Explorer</div>
                <div class="menu-sep"></div>
                <div>Select Program...</div>
            </div>
        </div>
        <div iconCls="icon-remove">Delete</div>
        <div>Select All</div>
    </div>
```

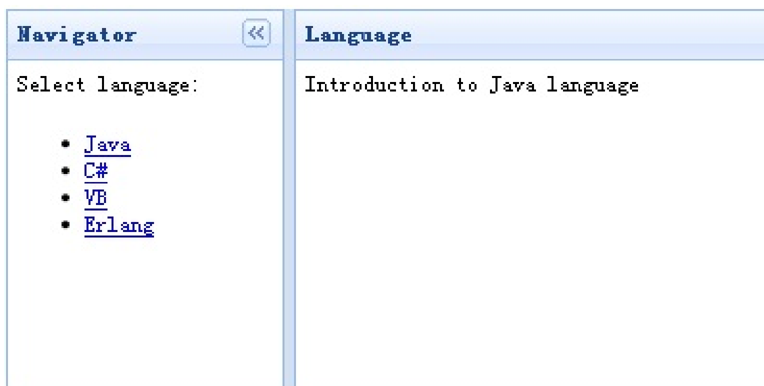
现在已经定义好了一个分割按钮（Split Button），您不需要写任何的 javascript 代码。

jQuery EasyUI 布局

jQuery EasyUI 布局 - 为网页创建边框布局

边框布局（border layout）提供五个区域：east、west、north、south、center。以下是一些通常用法：

- north 区域可以用来显示网站的标语。
- south 区域可以用来显示版权以及一些说明。
- west 区域可以用来显示导航菜单。
- east 区域可以用来显示一些推广的项目。
- center 区域可以用来显示主要的内容。



为了应用布局（layout），您应该确定一个布局（layout）容器，然后定义一些区域。布局（layout）必须至少需要一个 center 区域，以下是一个布局（layout）实例：



```
<div class="easyui-layout" style="width:400px;height:200px;">
    <div region="west" split="true" title="Navigator" style="w
        <p style="padding:5px;margin:0;">Select language:</p>
        <ul>
            <li><a href="javascript:void(0)" onclick="showconte
            <li><a href="javascript:void(0)" onclick="showconte
            <li><a href="javascript:void(0)" onclick="showconte
            <li><a href="javascript:void(0)" onclick="showconte
        </ul>
    </div>
    <div id="content" region="center" title="Language" style="p
    </div>
</div>
```

我们在一个 <div> 容器中创建了一个边框布局（border layout），布局（layout）把容器切割为两个部分，左边是导航菜单，右边是主要内容。

最后我们写一个 onclick 事件处理函数来检索数据，'showcontent' 函数非常简单：

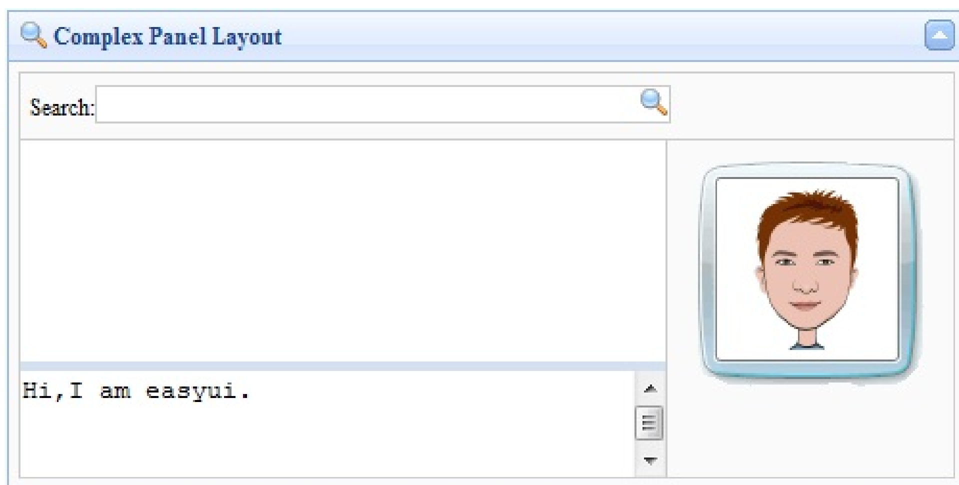
```
    function showcontent(language){
        $('#content').html('Introduction to ' + language + ' langua
    }
```

下载 jQuery EasyUI 实例

jeasyui-layout-layout.zip

jQuery EasyUI 布局 - 在面板中创建复杂布局

面板（Panel）允许您创建用于多种用途的自定义布局。在本实例中，我们使用面板（panel）和布局（layout）插件来创建一个 msn 消息框。



我们在区域面板中使用多个布局（layout）。在消息框的顶部我们放置一个查询输入框，同时在右边放置一个人物图片。在中间的区域我们通过设置 split 属性为 true，把这部分切割为两部分，允许用户改变区域面板的尺寸大小。

以下就是所有代码：

```
<div class="easyui-panel" title="Complex Panel Layout" iconCls=
    <div class="easyui-layout" fit="true">
        <div region="north" border="false" class="p-search">
            <label>Search:</label><input></input>
        </div>
        <div region="center" border="false">
            <div class="easyui-layout" fit="true">
                <div region="east" border="false" class="p-righ
                    <img src="images/msn.gif"/>
                </div>
                <div region="center" border="false" style="bor
                    <div class="easyui-layout" fit="true">
                        <div region="south" split="true" border
                            <textarea style="border:0;width:10
                        </div>
                        <div region="center" border="false">
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
```

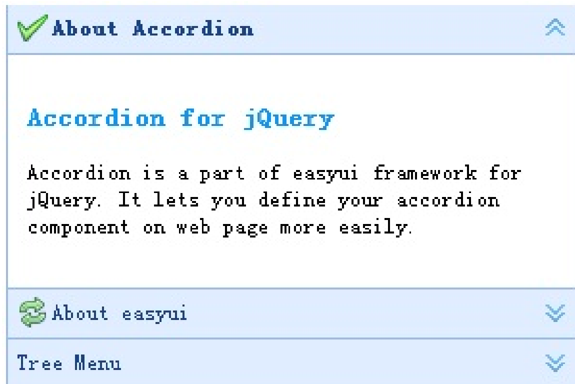
我们不需要写任何的 javascript 代码，它自己有非常强大的设计用户界面的功能。

下载 jQuery EasyUI 实例

jeasyui-layout-panel.zip

jQuery EasyUI 布局 - 创建折叠面板

在本教程中，您将会学习到关于 easyui 折叠面板（Accordion）的知识。 折叠面板（Accordion）包含一系列的面板（panel）。 所有面板（panel）的头部（header）都是可见的，但是一次仅仅显示一个面板（panel）的 body 内容。 当用户点击面板（panel）的头部（header）时，该面板（panel）的 body 内容将可见，同时其他面板（panel）的 body 内容将隐藏不可见。



我们将创建三个面板（panel），第三个面板（panel）包含一个树形菜单。

```
<div class="easyui-accordion" style="width:300px;height:200px;'
    <div title="About Accordion" iconCls="icon-ok" style="overf
        <h3 style="color:#0099FF;">Accordion for jQuery</h3>
        <p>Accordion is a part of easyui framework for jQuery.
    </div>
    <div title="About easyui" iconCls="icon-reload" selected="t
        easyui help you build your web page easily
    </div>
    <div title="Tree Menu">
        <ul id="tt1" class="easyui-tree">
            <li>
                <span>Folder1</span>
                <ul>
                    <li>
                        <span>Sub Folder 1</span>
                        <ul>
                            <li><span>File 11</span></li>
                            <li><span>File 12</span></li>
                            <li><span>File 13</span></li>
                        </ul>
                    </li>
                    <li><span>File 2</span></li>
                    <li><span>File 3</span></li>
                </ul>
            </li>
            <li><span>File2</span></li>
        </ul>
    </div>
</div>
```

下载 jQuery EasyUI 实例

jeasyui-layout-accordion.zip

jQuery EasyUI 布局 - 创建标签页（Tabs）

本教程将向您延时如何使用 easyui 创建一个 Tabs 组件。 Tabs 有多个可以动态地添加或移除的面板（panel）。 您可以使用 Tabs 来在相同的页面上显示不同的实体。

Tabs 一次仅仅显示一个面板（panel），每个面板（panel）都有标题、图标和关闭按钮。 当 Tabs 被选中时，将显示对应的面板（panel）的内容。



从 HTML 标记创建 Tabs，包含一个 DIV 容器和一些 DIV 面板（panel）。

```
<div class="easyui-tabs" style="width:400px;height:100px;">
    <div title="First Tab" style="padding:10px;">
        First Tab
    </div>
    <div title="Second Tab" closable="true" style="padding:10p
        Second Tab
    </div>
    <div title="Third Tab" iconCls="icon-reload" closable="true
        Third Tab
    </div>
</div>
```

我们创建一个带有三个面板（panel）的 Tabs 组件，第二个和第三个面板（panel）可以通过点击关闭按钮进行关闭。

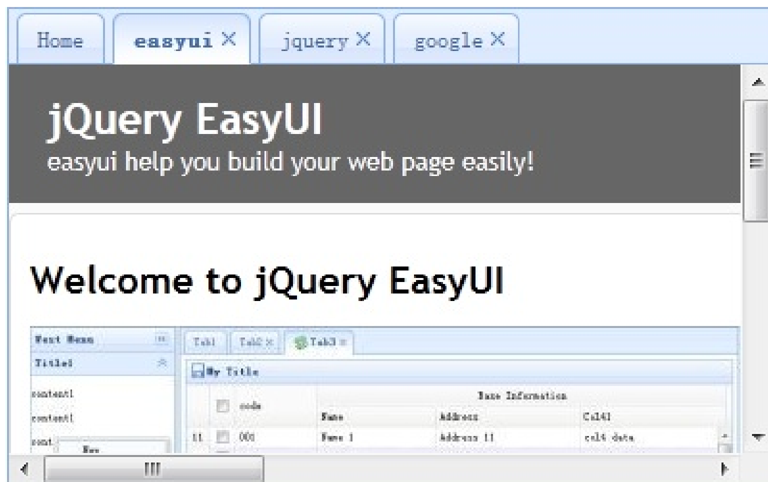
下载 jQuery EasyUI 实例

jeasyui-layout-tabs1.zip

jQuery EasyUI 布局 - 动态添加标签页（Tabs）

通过使用 jQuery EasyUI 可以很容易地添加 Tabs。您只需要调用 'add' 方法即可。

在本教程中，我们将使用 iframe 动态地添加显示在一个页面上的 Tabs。 当点击添加按钮，一个新的 tab 将被添加。如果 tab 已经存在，它将被激活。



步骤 1：创建 Tabs

```
<div style="margin-bottom:10px">
    <a href="#" class="easyui-linkbutton" onclick="addTab('goog
    <a href="#" class="easyui-linkbutton" onclick="addTab('jque
    <a href="#" class="easyui-linkbutton" onclick="addTab('easy
</div>
<div id="tt" class="easyui-tabs" style="width:400px;height:250p
    <div title="Home">
    </div>
</div>
```

这个 html 代码非常简单,我们创建了带有一个名为 'Home' 的 tab 面板的 Tabs。请注意，我们不需要写任何的 js 代码。

步骤 2：实现 'addTab' 函数

```
    function addTab(title, url){
        if ($('#tt').tabs('exists', title)){
            $('#tt').tabs('select', title);
        } else {
            var content = '<iframe scrolling="auto" frameborder="0'
            $('#tt').tabs('add',{
                title:title,
                content:content,
                closable:true
            });
        }
    }
```

我们使用 'exists' 方法来判断 tab 是否已经存在，如果已存在则激活 tab。如果不存在则调用 'add' 方法来添加一个新的 tab 面板。

下载 jQuery EasyUI 实例

jeasyui-layout-tabs2.zip

jQuery EasyUI 布局 - 添加自动播放标签页（Tabs）

本教程将向您展示如何创建一个自动播放的 Tabs。 Tabs 组件显示第一个 tab 面板，然后显示第二个、第三个... 我们将写一些代码来自动地切换 tab 面板，然后让它循环。



步骤 1：创建 Tabs

```
<div id="tt" class="easyui-tabs" style="width:330px;height:270p
    <div title="Shirt1" style="padding:20px;">
        <img src="images/shirt1.gif">
    </div>
    <div title="Shirt2" style="padding:20px;">
        <img src="images/shirt2.gif">
    </div>
    <div title="Shirt3" style="padding:20px;">
        <img src="images/shirt3.gif">
    </div>
    <div title="Shirt4" style="padding:20px;">
        <img src="images/shirt4.gif">
    </div>
    <div title="Shirt5" style="padding:20px;">
        <img src="images/shirt5.gif">
    </div>
</div>
```

步骤 2：写自动播放代码

```
    var index = 0;
    var t = $('#tt');
    var tabs = t.tabs('tabs');
    setInterval(function(){
        t.tabs('select', tabs[index].panel('options').title);
        index++;
        if (index >= tabs.length){
            index = 0;
        }
    }, 3000);
```

正如您所看到的，我们调用 'tabs' 方法来得到所有 tab 面板，并使用 'setInterval' 函数来切换他们。

下载 jQuery EasyUI 实例

jeasyui-layout-tabs3.zip

jQuery EasyUI 布局 - 创建 XP 风格左侧面板

通常情况下，在 Windows XP 的资源管理器文件夹中，左侧的面板（panel）包含一些常见任务。 本教程向您展示如何通过 easyui 的面板（panel）插件来创建 XP 左侧面板。



定义一些面板（panel）

我们定义一些面板（panel），这些面板（panel）用来显示一些任务。每个面板（panel）应该至少有折叠/展开工具按钮。

代码如下所示：

```
<div style="width:200px;height:auto;background:#7190E0;padding
    <div class="easyui-panel" title="Picture Tasks" collapsible
        View as a slide show<br/>
        Order prints online<br/>
        Print pictures
    </div>
    <br/>
    <div class="easyui-panel" title="File and Folder Tasks" col
        Make a new folder<br/>
        Publish this folder to the Web<br/>
        Share this folder
    </div>
    <br/>
    <div class="easyui-panel" title="Other Places" collapsible=
        New York<br/>
        My Pictures<br/>
        My Computer<br/>
        My Network Places
    </div>
    <br/>
    <div class="easyui-panel" title="Details" collapsible="true
        My documents<br/>
        File folder<br/><br/>
        Date modified: Oct.3rd 2010
    </div>
</div>
```

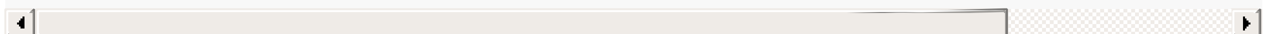


自定义面板（panel）的外观效果

请注意，这个视图外观效果不是我们想要的，我们必须改变面板（panel）的头部背景图片和折叠/展开按钮的图标。

做到这一点并不难，我们需要做的只是重新定义一些 CSS。

```
.panel-body{
    background:#f0f0f0;
}
.panel-header{
    background:#fff url('images/panel_header_bg.gif') no-repeat
}
.panel-tool-collapse{
    background:url('images/arrow_up.gif') no-repeat 0px -3px;
}
.panel-tool-expand{
    background:url('images/arrow_down.gif') no-repeat 0px -3px
}
```



由此可见，使用 easyui 定义用户界面非常简单。

下载 jQuery EasyUI 实例

jeasyui-layout-xp.zip

jQuery EasyUI 数据网格

jQuery EasyUI 数据网格 - 转换 HTML 表格为数据网格

本实例演示如何转换表格（table）为数据网格（datagrid）。

Col 1	Col 2	Col 3	Col 4	Col 5	Col 6
Data 1	Data 2	Data 3	Data 4	Data 5	Data 6
Data 1	Data 2	Data 3	Data 4	Data 5	Data 6
Data 1	Data 2	Data 3	Data 4	Data 5	Data 6
Data 1	Data 2	Data 3	Data 4	Data 5	Data 6

数据网格（datagrid）的列信息是定义在 <thead> 标记中，数据是定义在 <tbody> 标记中。确保为所有的数据列设置 field 名称，请看下面的实例：

```
<table id="tt" class="easyui-datagrid" style="width:400px;heigh
    <thead>
        <tr>
            <th field="name1" width="50">Col 1</th>
            <th field="name2" width="50">Col 2</th>
            <th field="name3" width="50">Col 3</th>
            <th field="name4" width="50">Col 4</th>
            <th field="name5" width="50">Col 5</th>
            <th field="name6" width="50">Col 6</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td>Data 1</td>
            <td>Data 2</td>
            <td>Data 3</td>
            <td>Data 4</td>
            <td>Data 5</td>
            <td>Data 6</td>
        </tr>
        <tr>
            <td>Data 1</td>
            <td>Data 2</td>
            <td>Data 3</td>
            <td>Data 4</td>
            <td>Data 5</td>
            <td>Data 6</td>
        </tr>
        <tr>
            <td>Data 1</td>
            <td>Data 2</td>
            <td>Data 3</td>
            <td>Data 4</td>
            <td>Data 5</td>
            <td>Data 6</td>
        </tr>
        <tr>
            <td>Data 1</td>
            <td>Data 2</td>
            <td>Data 3</td>
            <td>Data 4</td>
            <td>Data 5</td>
            <td>Data 6</td>
        </tr>
    </tbody>
</table>
```

非常棒，您可以定义一个复杂的表头，例如：

```
    <thead>
        <tr>
            <th field="name1" width="50" rowspan="2">Col 1</th>
            <th field="name2" width="50" rowspan="2">Col 2</th>
            <th field="name3" width="50" rowspan="2">Col 3</th>
            <th colspan="3">Details</th>
        </tr>
        <tr>
            <th field="name4" width="50">Col 4</th>
            <th field="name5" width="50">Col 5</th>
            <th field="name6" width="50">Col 6</th>
        </tr>
    </thead>
```

现在您可以看见，复杂表头已经创建。

Col 1	Col 2	Col 3	Details		
			Col 4	Col 5	Col 6
Data 1	Data 2	Data 3	Data 4	Data 5	Data 6
Data 1	Data 2	Data 3	Data 4	Data 5	Data 6
Data 1	Data 2	Data 3	Data 4	Data 5	Data 6
Data 1	Data 2	Data 3	Data 4	Data 5	Data 6

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid1.zip

jQuery EasyUI 数据网格 - 取得选中行数据

本实例演示如何取得选中行数据。

Load Data

Item ID	Product ID	List Price	Unit Cost	Attribute	Status
EST-1	FI-SW-01	16.5	10	Large	P
EST-10	K9-DL-01	18.5	12	Spotted Adult Female	P
EST-11	RP-SN-01	18.5	12	Venomless	P
EST-12	RP-SN-01	18.5	12	Rattleless	P
EST-13	RP-LI-02	18.5	12	Green Adult	P
EST-14	FL-DSH-01	58.5	12	Tailless	P
EST-15	FL-DSH-01	23.5	12	With tail	P
EST-16	FL-DLH-02	93.5	12	Adult Female	P
EST-17	FL-DLH-02	93.5	12	Adult Male	P
EST-18	AV-CB-01	193.5	92	Adult Male	P

数据网格（datagrid）组件包含两种方法来检索选中行数据：

- getSelected：取得第一个选中行数据，如果没有选中行，则返回 null，否则返回记录。
- getSelections：取得所有选中行数据，返回元素记录的数组数据。

创建数据网格（DataGrid）

```
<table id="tt" class="easyui-datagrid" style="width:600px;heigh
        url="data/datagrid_data.json"
        title="Load Data" iconCls="icon-save">
    <thead>
        <tr>
            <th field="itemid" width="80">Item ID</th>
            <th field="productid" width="80">Product ID</th>
            <th field="listprice" width="80" align="right">List
            <th field="unitcost" width="80" align="right">Unit
            <th field="attr1" width="150">Attribute</th>
            <th field="status" width="60" align="center">Stauts
        </tr>
    </thead>
</table>
```

使用演示

取得选中行数据：

```
var row = $('#tt').datagrid('getSelected');
if (row){
    alert('Item ID:'+row.itemid+"\nPrice:"+row.listprice);
}
```

取得所有选中行的 itemid：

```
var ids = [];
var rows = $('#tt').datagrid('getSelections');
for(var i=0; i<rows.length; i++){
    ids.push(rows[i].itemid);
}
alert(ids.join('\n'));
```

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid3.zip

jQuery EasyUI 数据网格 - 添加查询功能

本实例演示如何从数据库得到数据，并将它们显示在数据网格（datagrid）中。然后演示如何根据用户输入的搜索关键词搜寻显示结果。

Searching

Item ID: Product ID: K Search

	Item ID	Product ID	List Price	Unit Cost	Attribute	Stauts
3	EST-23	K9-RT-02	145.49	100.00	Adult Female	P
4	EST-24	K9-RT-02	255.50	92.00	Adult Male	P
5	EST-25	K9-RT-02	325.29	90.00	Adult Female	P
6	EST-26	K9-CW-01	125.50	92.00	Adult Male	P
7	EST-27	K9-CW-01	155.29	90.00	Adult Female	P
8	EST-28	K9-RT-01	155.29	90.00	Adult Female	P

10 Page 1 of 2 Displaying 1 to 10 of 12 items

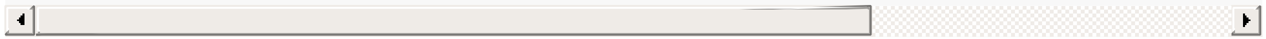
创建数据网格（DataGrid）

创建带有分页功能的数据网格（datagrid），然后添加工具栏到其中。

```
    <table id="tt" class="easyui-datagrid" style="width:600px;heigh
            url="datagrid24_getdata.php" toolbar="#tb"
            title="Load Data" iconCls="icon-save"
            rownumbers="true" pagination="true">
        <thead>
            <tr>
                <th field="itemid" width="80">Item ID</th>
                <th field="productid" width="80">Product ID</th>
                <th field="listprice" width="80" align="right">List
                <th field="unitcost" width="80" align="right">Unit
                <th field="attr1" width="150">Attribute</th>
                <th field="status" width="60" align="center">Stauts
            </tr>
        </thead>
    </table>
```

工具栏定义如下：

```
<div id="tb" style="padding:3px">
    <span>Item ID:</span>
    <input id="itemid" style="line-height:26px;border:1px solid
    <span>Product ID:</span>
    <input id="productid" style="line-height:26px;border:1px so
    <a href="#" class="easyui-linkbutton" plain="true" onclick=
</div>
```



当用户输入查询值并按下查询按钮时，'doSearch' 函数将被调用：

```
function doSearch(){
    $('#tt').datagrid('load',{
        itemid: $('#itemid').val(),
        productid: $('#productid').val()
    });
}
```

上面的代码调用了 'load' 方法来加载新的数据网格（datagrid）数据。我们需要传递 'itemid' 和 'productid' 参数到服务器。

服务器端代码

```
include 'conn.php';

$page = isset($_POST['page']) ? intval($_POST['page']) : 1;
$rows = isset($_POST['rows']) ? intval($_POST['rows']) : 10;
$itemid = isset($_POST['itemid']) ? mysql_real_escape_string($_
$productid = isset($_POST['productid']) ? mysql_real_escape_str

$offset = ($page-1)*$rows;

$result = array();

$where = "itemid like '$itemid%' and productid like '$productic
$rs = mysql_query("select count(*) from item where " . $where);
$row = mysql_fetch_row($rs);
$result["total"] = $row[0];

$rs = mysql_query("select * from item where " . $where . " lim

$items = array();
while($row = mysql_fetch_object($rs)){
    array_push($items, $row);
}
$result["rows"] = $items;

echo json_encode($result);
```

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid24.zip

jQuery EasyUI 数据网格 - 添加工具栏

本实例演示如何添加工具栏（toolbar）到数据网格（datagrid）。

DataGrid with Toolbar

Add Cut Save

Item ID	Product ID	List Price	Unit Cost	Attribute	Status
EST-1	FI-SW-01	16.5	10	Large	P
EST-10	K9-DL-01	18.5	12	Spotted Adult Fe	P
EST-11	RP-SN-01	18.5	12	Venomless	P
EST-12	RP-SN-01	18.5	12	Rattleless	P
EST-13	RP-LI-02	18.5	12	Green Adult	P
EST-14	FL-DSH-01	58.5	12	Tailless	P
EST-15	FL-DSH-01	23.5	12	With tail	P
EST-16	FL-DLH-02	93.5	12	Adult Female	P

创建数据网格（DataGrid）

```
    <table id="tt" class="easyui-datagrid" style="width:600px;heigh
            url="data/datagrid_data.json"
            title="DataGrid with Toolbar" iconCls="icon-save"
            toolbar="#tb">
        <thead>
            <tr>
                <th field="itemid" width="80">Item ID</th>
                <th field="productid" width="80">Product ID</th>
                <th field="listprice" width="80" align="right">List
                <th field="unitcost" width="80" align="right">Unit
                <th field="attr1" width="150">Attribute</th>
                <th field="status" width="60" align="center">Stauts
            </tr>
        </thead>
    </table>
    <div id="tb">
        <a href="#" class="easyui-linkbutton" iconCls="icon-add" pl
        <a href="#" class="easyui-linkbutton" iconCls="icon-cut" pl
        <a href="#" class="easyui-linkbutton" iconCls="icon-save" p
    </div>
```

我们不需要写任何的 javascript 代码，只需通过 'toolbar' 属性附加工具栏（toolbar）到数据网格（datagrid）。

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid4.zip

jQuery EasyUI 数据网格 - 创建复杂工具栏

数据网格（datagrid）的工具栏（toolbar）可以包含按钮及其他组件。 您可以通个一个已存在的 DIV 标签来简单地定义工具栏布局，该 DIV 标签将成为数据网格（datagrid）工具栏的内容。 本教程将向您展示如何创建数据网格（datagrid）组件的复杂工具栏。



创建工具栏（Toolbar）

```
<div id="tb" style="padding:5px;height:auto">
    <div style="margin-bottom:5px">
        <a href="#" class="easyui-linkbutton" iconCls="icon-add
        <a href="#" class="easyui-linkbutton" iconCls="icon-edi
        <a href="#" class="easyui-linkbutton" iconCls="icon-sav
        <a href="#" class="easyui-linkbutton" iconCls="icon-cut
        <a href="#" class="easyui-linkbutton" iconCls="icon-rem
    </div>
    <div>
        Date From: <input class="easyui-datebox" style="width:8
        To: <input class="easyui-datebox" style="width:80px">
        Language:
        <input class="easyui-combobox" style="width:100px"
                url="data/combobox_data.json"
                valueField="id" textField="text">
        <a href="#" class="easyui-linkbutton" iconCls="icon-sea
    </div>
</div>
```

创建数据网格（DataGrid）

```
<table class="easyui-datagrid" style="width:600px;height:250px'
        url="data/datagrid_data.json"
        title="DataGrid - Complex Toolbar" toolbar="#tb"
        singleSelect="true" fitColumns="true">
    <thead>
        <tr>
            <th field="itemid" width="60">Item ID</th>
            <th field="productid" width="80">Product ID</th>
            <th field="listprice" align="right" width="70">List
            <th field="unitcost" align="right" width="70">Unit
            <th field="attr1" width="200">Address</th>
            <th field="status" width="50">Status</th>
        </tr>
    </thead>
</table>
```

正如您所看到的，数据网格（datagrid）的工具栏域对话框（dialog）相似。我们不需要写任何的 javascript 代码，就能创建带有复杂工具栏的数据网格（datagrid）。

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid19.zip

jQuery EasyUI 数据网格 - 设置冻结列

本实例演示如何冻结一些列，当用户在网格上移动水平滚动条时，冻结列不能滚动到视图的外部。

Frozen Columns

Item ID	Product ID	List Price	Unit Cost	Attribute	Status
EST-1	FI-SW-01	16.5	10	Large	P
EST-10	K9-DL-01	18.5	12	Spotted Adult Fe	P
EST-11	RP-SN-01	18.5	12	Venomless	P
EST-12	RP-SN-01	18.5	12	Rattleless	P
EST-13	RP-LI-02	18.5	12	Green Adult	P
EST-14	FL-DSH-01	58.5	12	Tailless	P
EST-15	FL-DSH-01	23.5	12	With tail	P
EST-16	FL-DLH-02	93.5	12	Adult Female	P
EST-17	FL-DLH-02	93.5	12	Adult Male	P
EST-18	AV-CB-01				

为了冻结列，您需要定义 frozenColumns 属性。frozenColumn 属性和 columns 属性一样。

```
$('#tt').datagrid({
    title:'Frozen Columns',
    iconCls:'icon-save',
    width:500,
    height:250,
    url:'data/datagrid_data.json',
    frozenColumns:[[
        {field:'itemid',title:'Item ID',width:80},
        {field:'productid',title:'Product ID',width:80},
    ]],
    columns:[[
        {field:'listprice',title:'List Price',width:80,align:'r
        {field:'unitcost',title:'Unit Cost',width:80,align:'ri
        {field:'attr1',title:'Attribute',width:100},
        {field:'status',title:'Status',width:60}
    ]]
});
```

您不需要写任何的 javascript 代码，这样您就能创建一个数据网格（datagrid）组件，如下所示：

```
<table id="tt" title="Frozen Columns" class="easyui-datagrid" s
        url="data/datagrid_data.json"
        singleSelect="true" iconCls="icon-save">
    <thead frozen="true">
        <tr>
            <th field="itemid" width="80">Item ID</th>
            <th field="productid" width="80">Product ID</th>
        </tr>
    </thead>
    <thead>
        <tr>
            <th field="listprice" width="80" align="right">List
            <th field="unitcost" width="80" align="right">Unit
            <th field="attr1" width="150">Attribute</th>
            <th field="status" width="60" align="center">Stauts
        </tr>
    </thead>
</table>
```

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid5.zip

jQuery EasyUI 数据网格 - 动态改变列

数据网格（DataGrid）列可以使用 'columns' 属性简单地定义。如果您想动态地改变列，那根本没有问题。为了改变列，您可以重新调用datagrid 方法，并传递一个新的 columns 属性。

创建数据网格（DataGrid）

```
<table id="tt" title="Frozen Columns" class="easyui-datagrid" s
        url="data/datagrid_data.json"
        singleSelect="true" iconCls="icon-save">
</table>
```

```
$('#tt').datagrid({
    columns:[[
        {field:'itemid',title:'Item ID',width:80},
        {field:'productid',title:'Product ID',width:80},
        {field:'attr1',title:'Attribute',width:200},
        {field:'status',title:'Status',width:80}
    ]]
});
```

运行网页，您将看见：

Change Columns

Item ID	Product ID	Attribute	Status
EST-1	FI-SW-01	Large	P
EST-10	K9-DL-01	Spotted Adult Female	P
EST-11	RP-SN-01	Venomless	P
EST-12	RP-SN-01	Rattleless	P
EST-13	RP-LI-02	Green Adult	P
EST-14	FL-DSH-01	Tailless	P
EST-15	FL-DSH-01	With tail	P
EST-16	FL-DLH-02	Adult Female	P
EST-17	FL-DLH-02	Adult Male	P

可是有时候您想改变列，所以您需要写一些代码：

```
$('#tt').datagrid({
    columns:[[
        {field:'itemid',title:'Item ID',width:80},
        {field:'productid',title:'Product ID',width:80},
        {field:'listprice',title:'List Price',width:80,align:'r
        {field:'unitcost',title:'Unit Cost',width:80,align:'rig
        {field:'attr1',title:'Attribute',width:100},
        {field:'status',title:'Status',width:60}
    ]]
});
```

请记住，我们已经定义了其他属性，比如：url、width、height 等等。我们不需要再一次定义它们，我们定义那些我们需要改变的。

Change Columns

Item ID	Product ID	List Price	Unit Cost	Attribute	Status
EST-1	FI-SW-01	16.5	10	Large	P
EST-10	K9-DL-01	18.5	12	Spotted Adult Fe	P
EST-11	RP-SN-01	18.5	12	Venomless	P
EST-12	RP-SN-01	18.5	12	Rattleless	P
EST-13	RP-LI-02	18.5	12	Green Adult	P
EST-14	FL-DSH-01	58.5	12	Tailless	P
EST-15	FL-DSH-01	23.5	12	With tail	P
EST-16	FL-DLH-02	93.5	12	Adult Female	P
EST-17	FL-DLH-02	93.5	12	Adult Male	P

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid6.zip

jQuery EasyUI 数据网格 - 格式化列

以下实例格式化在 easyui DataGrid 里的列数据，并使用自定义列的 formatter，如果价格小于 20 就将文本变为红色。

Formatting Columns

Item ID	Product ID	List Price	Unit Cost	Attribute	Status
EST-1	FI-SW-01	(16.5)	10	Large	P
EST-10	K9-DL-01	(18.5)	12	Spotted Adult Fe	P
EST-11	RP-SN-01	(18.5)	12	Venomless	P
EST-12	RP-SN-01	(18.5)	12	Rattleless	P
EST-13	RP-LI-02	(18.5)	12	Green Adult	P
EST-14	FL-DSH-01	58.5	12	Tailless	P
EST-15	FL-DSH-01	23.5	12	With tail	P
EST-16	FL-DLH-02	93.5	12	Adult Female	P
EST-17	FL-DLH-02	93.5	12	Adult Male	P

为了格式化一个数据网格（DataGrid）列，我们需要设置 formatter 属性，它是一个函数。这个格式化函数包含三个参数：

- value：当前列对应字段值。
- row：当前的行记录数据。
- index：当前的行下标。

创建数据网格（DataGrid）

```
<table id="tt" title="Formatting Columns" class="easyui-datagri
        url="data/datagrid_data.json"
        singleSelect="true" iconCls="icon-save">
    <thead>
        <tr>
            <th field="itemid" width="80">Item ID</th>
            <th field="productid" width="80">Product ID</th>
            <th field="listprice" width="80" align="right" forr
            <th field="unitcost" width="80" align="right">Unit
            <th field="attr1" width="100">Attribute</th>
            <th field="status" width="60" align="center">Stauts
        </tr>
    </thead>
</table>
```

请注意，'listprice' 字段有一个 'formatter' 属性，用来指明格式化函数。

写格式化函数

```
function formatPrice(val,row){
    if (val < 20){
        return '<span style="color:red;">('+val+')</span>';
    } else {
        return val;
    }
}
```

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid7.zip

jQuery EasyUI 数据网格 - 设置排序

本实例演示如何通过点击列表头来排序数据网格（DataGrid）。

Load Data

	Item ID	Product ID	List Price ▲	Unit Cost	Attribute	Stauts
21	EST-26	K9-CW-01	125.50	92.00	Adult Male	P
22	EST-22	K9-RT-02	135.50	100.00	Adult Male	P
23	EST-23	K9-RT-02	145.49	100.00	Adult Female	P
24	EST-28	K9-RT-01	155.29	90.00	Adult Female	P
25	EST-27	K9-CW-01	155.29	90.00	Adult Female	P
26	EST-18	AV-CB-01	193.50	92.00	Adult Male	P
27	EST-24	K9-RT-02	255.50	92.00	Adult Male	P
28	EST-25	K9-RT-02	325.29	90.00	Adult Female	P

10 Page 3 of 3 Displaying 21 to 28 of 28 items

数据网格（DataGrid）的所有列可以通过点击列表头来排序。您可以定义哪列可以排序。默认的，列是不能排序的，除非您设置 sortable 属性为 true。

创建数据网格（DataGrid）

```
<table id="tt" class="easyui-datagrid" style="width:600px;heigh
        url="datagrid8_getdata.php"
        title="Load Data" iconCls="icon-save"
        rownumbers="true" pagination="true">
    <thead>
        <tr>
            <th field="itemid" width="80" sortable="true">Item 
            <th field="productid" width="80" sortable="true">Pr
            <th field="listprice" width="80" align="right" sort
            <th field="unitcost" width="80" align="right" sorta
            <th field="attr1" width="150">Attribute</th>
            <th field="status" width="60" align="center">Stauts
        </tr>
    </thead>
</table>
```

我们定义一些可排序的列，包含 itemid、productid、listprice、unitcost 等等。'attr1' 列和 'status' 列不能排序。

当排序时，数据网格（DataGrid）将发送两个参数到远程服务器：

- sort：排序列字段名。
- order：排序方式，可以是 'asc' 或者 'desc'，默认值是 'asc'。

服务器端代码

```
$page = isset($_POST['page']) ? intval($_POST['page']) : 1;
$rows = isset($_POST['rows']) ? intval($_POST['rows']) : 10;
$sort = isset($_POST['sort']) ? strval($_POST['sort']) : 'item
$order = isset($_POST['order']) ? strval($_POST['order']) : 'as
$offset = ($page-1)*$rows;

$result = array();

include 'conn.php';

$rs = mysql_query("select count(*) from item");
$row = mysql_fetch_row($rs);
$result["total"] = $row[0];

$rs = mysql_query("select * from item order by $sort $order lin

$items = array();
while($row = mysql_fetch_object($rs)){
    array_push($items, $row);
}
$result["rows"] = $items;

echo json_encode($result);
```

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid8.zip

jQuery EasyUI 数据网格 - 自定义排序

如果默认的排序行为不满足您的需求，您可以自定义数据网格（datagrid）的排序行为。

Custom Sort

Item ID	List Price	Unit Cost	Attribute	Date ▲	Status
EST-15	23.5	12	With tail	11/04/2005	P
EST-11	18.5	12	Venomless	12/31/2005	P
EST-18	193.5	92	Adult Male	03/04/2006	P
EST-17	93.5	12	Adult Male	03/04/2008	P
EST-10	18.5	12	Spotted Adult Female	05/04/2009	P
EST-1	16.5	10	Large	03/04/2010	P
EST-12	18.5	12	Rattleless	03/21/2010	P
EST-16	93.5	12	Adult Female	03/23/2010	P
EST-14	58.5	12	Tailless	07/04/2010	P

最基础的，用户可以在列上定义一个排序函数，函数名是 sorter。这个函数将接受两个值，返回值将如下：

valueA > valueB => 返回 1

valueA < valueB => 返回 -1

自定义排序代码

```
<table id="tt"></table>
```

```
$('#tt').datagrid({
    title:'Custom Sort',
    iconCls:'icon-ok',
    width:520,
    height:250,
    singleSelect:true,
    remoteSort:false,
    columns:[[
        {field:'itemid',title:'Item ID',width:60,sortable:true]
        {field:'listprice',title:'List Price',width:70,align:'r
        {field:'unitcost',title:'Unit Cost',width:70,align:'ri
        {field:'attr1',title:'Attribute',width:120,sortable:tru
        {field:'date',title:'Date',width:80,sortable:true,align
            sorter:function(a,b){
                a = a.split('/');
                b = b.split('/');
                if (a[2] == b[2]){
                    if (a[0] == b[0]){
                        return (a[1]>b[1]?1:-1);
                    } else {
                        return (a[0]>b[0]?1:-1);
                    }
                } else {
                    return (a[2]>b[2]?1:-1);
                }
            }
        },
        {field:'status',title:'Status',width:40,align:'center']
    ]]
}).datagrid('loadData', data);
```

您可以从这段代码中看到，我们为 date 列创建了自定义的 sorter。日期的格式是 'dd/mm/yyyy'，可以轻松的按年月日排序。

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid14.zip

jQuery EasyUI 数据网格 - 创建列组合

easyui 的数据网格（DataGrid）可以创建列组合，如下所示：

Column Group

	Item ID	Product ID	Item Details			
			List Price	Unit Cost	Attribute	Status
1	EST-1	FI-SW-01	16.5	10	Large	P
2	EST-10	K9-DL-01	18.5	12	Spotted Adult Femal	P
3	EST-11	RP-SN-01	18.5	12	Venomless	P
4	EST-12	RP-SN-01	18.5	12	Rattleless	P
5	EST-13	RP-LI-02	18.5	12	Green Adult	P
6	EST-14	FL-DSH-01	58.5	12	Tailless	P
7	EST-15	FL-DSH-01	23.5	12	With tail	P
8	EST-16	FL-DLH-02	93.5	12	Adult Female	P

在本实例中，我们使用平面数据来填充数据网格（DataGrid）的数据，并把 listprice、unitcost、addr1、status 列组合在一个单一的列下。

为了创建列组合，您应该定义数据网格（datagrid）插件的 columns 数据。列的每个元素是定义一组可使用 rowspan 或 colspan 属性来进行组合的单元格。

下面的代码实现了上面的实例：

```
<table id="tt" title="Column Group" class="easyui-datagrid" sty
        url="data/datagrid_data.json"
        singleSelect="true" iconCls="icon-save" rownumbers="tru
    <thead>
        <tr>
            <th rowspan="2" field="itemid" width="80">Item ID</
            <th rowspan="2" field="productid" width="80">Produc
            <th colspan="4">Item Details</th>
        </tr>
        <tr>
            <th field="listprice" width="80" align="right">List
            <th field="unitcost" width="80" align="right">Unit
            <th field="attr1" width="100">Attribute</th>
            <th field="status" width="60" align="center">Stauts
        </tr>
    </thead>
</table>
```

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid9.zip

jQuery EasyUI 数据网格 - 添加复选框

本实例演示如何放置一个复选框列到数据网格（DataGrid）。通过复选框，用户将可以选择 选中/取消选中 网格行数据。

Checkbox Select

☐	Item ID	Product ID	List Price	Unit Cost	Attribute	Status
☐	EST-1	FI-SW-01	16.5	10	Large	P
☐	EST-10	K9-DL-01	18.5	12	Spotted Adult Female	P
☐	EST-11	RP-SN-01	18.5	12	Venomless	P
☑	EST-12	RP-SN-01	18.5	12	Rattleless	P
☑	EST-13	RP-LI-02	18.5	12	Green Adult	P
☐	EST-14	FL-DSH-01	58.5	12	Tailless	P
☐	EST-15	FL-DSH-01	23.5	12	With tail	P
☐	EST-16	FL-DLH-02	93.5	12	Adult Female	P

10 | Page 1 of 3 | Displaying 1 to 10 of 28 items

为了添加一个复选框列，我们仅仅需要添加一个列的 checkbox 属性，并设置它为 true。代码如下所示：

```
<table id="tt" title="Checkbox Select" class="easyui-datagrid"
        url="data/datagrid_data.json"
        idField="itemid" pagination="true"
        iconCls="icon-save">
    <thead>
        <tr>
            <th field="ck" checkbox="true"></th>
            <th field="itemid" width="80">Item ID</th>
            <th field="productid" width="80">Product ID</th>
            <th field="listprice" width="80" align="right">List
            <th field="unitcost" width="80" align="right">Unit
            <th field="attr1" width="100">Attribute</th>
            <th field="status" width="60" align="center">Status
        </tr>
    </thead>
</table>
```

以上代码添加了一个带有 checkbox 属性的列，所以它将成为复选框列。如果 idField 属性已设置，数据网格（DataGrid）的选择集合将在不同的页面保持。

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid10.zip

jQuery EasyUI 数据网格 - 自定义分页

数据网格（datagrid）内置一个很好特性的分页功能，自定义也相当简单。在本教程中，我们将创建一个数据网格（datagrid），并在分页工具栏上添加一些自定义按钮。

Load Data

Item ID	Product ID	List Price	Unit Cost	Attribute	Status
EST-1	FI-SW-01	16.5	10	Large	P
EST-10	K9-DL-01	18.5	12	Spotted Adult Female	P
EST-11	RP-SN-01	18.5	12	Venomless	P
EST-12	RP-SN-01	18.5	12	Rattleless	P
EST-13	RP-LI-02	18.5	12	Green Adult	P
EST-14	FL-DSH-01	58.5	12	Tailless	P
EST-15	FL-DSH-01	23.5	12	With tail	P
EST-16	FL-DLH-02	93.5	12	Adult Female	P

Page 1 of 3　　Displaying 1 to 10 of 28 items

创建数据网格（DataGrid）

```
<table id="tt" title="Load Data" class="easyui-datagrid" style=
        url="data/datagrid_data.json"
        iconCls="icon-save" pagination="true">
    <thead>
        <tr>
            <th field="itemid" width="80">Item ID</th>
            <th field="productid" width="80">Product ID</th>
            <th field="listprice" width="80" align="right">List
            <th field="unitcost" width="80" align="right">Unit
            <th field="attr1" width="100">Attribute</th>
            <th field="status" width="60" align="center">Stauts
        </tr>
    </thead>
</table>
```

请记得，设置 'pagination' 属性为 true，这样才会生成分页工具栏。

自定义分页工具栏

```
var pager = $('#tt').datagrid('getPager');    // get the pager
pager.pagination({
    showPageList:false,
    buttons:[{
        iconCls:'icon-search',
        handler:function(){
            alert('search');
        }
    },{
        iconCls:'icon-add',
        handler:function(){
            alert('add');
        }
    },{
        iconCls:'icon-edit',
        handler:function(){
            alert('edit');
        }
    }],
    onBeforeRefresh:function(){
        alert('before refresh');
        return true;
    }
});
```

正如您所看到的，我们首先得到数据网格（datagrid）的 pager 对象，然后重新创建分页（pagination）。我们隐藏页面列表，并添加三个新的按钮。

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid11.zip

jQuery EasyUI 数据网格 - 启用行内编辑

可编辑的功能是最近添加到数据网格（datagrid）的。它可以使用户添加一个新行到数据网格（datagrid）。用户也可以更新一个或多个行。

本教程向您展示如何创建一个数据网格（datagrid）和内联编辑器。

Editable DataGrid

Item ID	Product	List Price	Unit Cost	Attribute	Status	Action
EST-1	Koi	16.5	10	Large	P	Edit Delete
EST-10	Dalmation	18.5	12	Spotted Adult Female	P	Edit Delete
EST-11	Rattlesnake	18.5	12	Venomless	☑	Save Cancel
EST-12	Rattlesnake	18.5	12	Rattleless	P	Edit Delete
EST-13	This field is required.			Green Adult	☑	Save Cancel
EST-14	Koi	58.5	12	Tailless	P	Edit Delete
EST-15	Dalmation	23.5	12	With tail	P	Edit Delete
EST-16	Rattlesnake	93.5	12	Adult Female	P	Edit Delete
EST-17	Iguana	93.5	12	Adult Male	P	Edit Delete

Manx
Persian
Amazon Parrot

创建数据网格（DataGrid）

```
    $(function(){
        $('#tt').datagrid({
            title:'Editable DataGrid',
            iconCls:'icon-edit',
            width:660,
            height:250,
            singleSelect:true,
            idField:'itemid',
            url:'datagrid_data.json',
            columns:[[
                {field:'itemid',title:'Item ID',width:60},
                {field:'productid',title:'Product',width:100,
                    formatter:function(value){
                        for(var i=0; i<products.length; i++){
                            if (products[i].productid == value) ret
                        }
                        return value;
                    },
                    editor:{
                        type:'combobox',
```

```
                        options:{
                            valueField:'productid',
                            textField:'name',
                            data:products,
                            required:true
                        }
                    }
                },
                {field:'listprice',title:'List Price',width:80,alig
                {field:'unitcost',title:'Unit Cost',width:80,align:
                {field:'attr1',title:'Attribute',width:150,editor:
                {field:'status',title:'Status',width:50,align:'cent
                    editor:{
                        type:'checkbox',
                        options:{
                            on: 'P',
                            off: ''
                        }
                    }
                },
                {field:'action',title:'Action',width:70,align:'cent
                    formatter:function(value,row,index){
                        if (row.editing){
                            var s = '<a href="#" onclick="saverow(t
                            var c = '<a href="#" onclick="cancelrow
                            return s+c;
                        } else {
                            var e = '<a href="#" onclick="editrow(t
                            var d = '<a href="#" onclick="deleterow
                            return e+d;
                        }
                    }
                }
            ]],
            onBeforeEdit:function(index,row){
                row.editing = true;
                updateActions(index);
            },
            onAfterEdit:function(index,row){
                row.editing = false;
                updateActions(index);
            },
            onCancelEdit:function(index,row){
                row.editing = false;
                updateActions(index);
            }
        });
    });
    function updateActions(index){
        $('#tt').datagrid('updateRow',{
            index: index,
            row:{}
        });
```

```
    }
```

为了启用数据网格行内编辑，您应该添加一个 editor 属性到列中。编辑器（editor）会告诉数据网格（datagrid）如何编辑字段及如何保存字段值。正如您所看到的，我们定义的三个编辑器（editor）：text、combobox 和 checkbox。

```
    function getRowIndex(target){
        var tr = $(target).closest('tr.datagrid-row');
        return parseInt(tr.attr('datagrid-row-index'));
    }
    function editrow(target){
        $('#tt').datagrid('beginEdit', getRowIndex(target));
    }
    function deleterow(target){
        $.messager.confirm('Confirm','Are you sure?',function(r){
            if (r){
                $('#tt').datagrid('deleteRow', getRowIndex(target)
            }
        });
    }
    function saverow(target){
        $('#tt').datagrid('endEdit', getRowIndex(target));
    }
    function cancelrow(target){
        $('#tt').datagrid('cancelEdit', getRowIndex(target));
    }
```

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid12.zip

jQuery EasyUI 数据网格 - 扩展编辑器

一些常见的编辑器（editor）添加到数据网格（datagrid），以便用户编辑数据。 所有的编辑器（editor）都定义在 $.fn.datagrid.defaults.editors 对象中，这个可以继承扩展以便支持新的编辑器（editor）。 本教程将向您展示如何添加一个新的 numberspinner 编辑器到数据网格（datagrid）。

Editable DataGrid

Item ID	List Price	Unit Cost	Attribute	Status	Action
EST-1	16.5	10	Large	P	Edit Delete
EST-10	18.5	12	Spotted Adult Female	P	Edit Delete
EST-11	18.5	12	Venomless	P	Edit Delete
EST-12	18.5	12	Rattleless	☑	Save Cancel
EST-13	18.5	12	Green Adult	P	Edit Delete
EST-14	58.5	12	Tailless	P	Edit Delete
EST-15	23.5	12	With tail	P	Edit Delete
EST-16	93.5	12	Adult Female	P	Edit Delete
EST-17	93.5	12	Adult Male	P	Edit Delete

继承扩展 numberspinner 编辑器

```
$.extend($.fn.datagrid.defaults.editors, {
    numberspinner: {
        init: function(container, options){
            var input = $('<input type="text">').appendTo(conta
            return input.numberspinner(options);
        },
        destroy: function(target){
            $(target).numberspinner('destroy');
        },
        getValue: function(target){
            return $(target).numberspinner('getValue');
        },
        setValue: function(target, value){
            $(target).numberspinner('setValue',value);
        },
        resize: function(target, width){
            $(target).numberspinner('resize',width);
        }
    }
});
```

在 html 标记中创建数据网格（DataGrid）

```
<table id="tt" style="width:600px;height:250px"
        url="data/datagrid_data.json" title="Editable DataGrid'
        singleSelect="true" idField="itemid" fitColumns="true">
    <thead>
        <tr>
            <th field="itemid" width="60">Item ID</th>
            <th field="listprice" width="80" align="right" edit
            <th field="unitcost" width="80" align="right" edito
            <th field="attr1" width="180" editor="text">Attribu
            <th field="status" width="60" align="center" editor
            <th field="action" width="80" align="center" format
        </tr>
    </thead>
</table>
```

我们分配 numberspinner 编辑器到 'unit cost' 字段。 当开始编辑一行，用户可以通过 numberspinner 编辑器来编辑数据。

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid23.zip

jQuery EasyUI 数据网格 - 列运算

在本教程中，您将学习如何在可编辑的数据网格（datagrid）中包含一个运算的列。一个运算列通常包含一些从一个或多个其他列运算的值。

Editable DataGrid with Calculated Column

Item ID	List Price	Amount	Unit Cost	Attribute	Status
EST-1	16.5		10	Large	P
EST-10	18.5		12	Spotted Adult Female	P
EST-11	18.5		12	Venomless	P
EST-12	18.5		12	Rattleless	P
EST-13	18.5	2	37	Green Adult	P
EST-14	58.5	2	117	Tailless	☑
EST-15	23.5		12	With tail	P
EST-16	93.5		12	Adult Female	P
EST-17	93.5		12	Adult Male	P
EST-18	193.5		92	Adult Male	P

首先，创建一个可编辑的数据网格（datagrid）。这里我们创建了一些可编辑列，'listprice'、'amount' 和 'unitcost' 列定义为 numberbox 编辑类型。运算列是 'unitcost' 字段，将是 listprice 乘以 amount 列的结果。

```
<table id="tt" style="width:600px;height:auto"
        title="Editable DataGrid with Calculated Column" iconCl
        idField="itemid" url="data/datagrid_data.json">
    <thead>
        <tr>
            <th field="itemid" width="80">Item ID</th>
            <th field="listprice" width="80" align="right" edit
            <th field="amount" width="80" align="right" editor=
            <th field="unitcost" width="80" align="right" edito
            <th field="attr1" width="150" editor="text">Attribu
            <th field="status" width="60" align="center" editor
        </tr>
    </thead>
</table>
```

当用户点击一行的时候，我们开始一个编辑动作。

```
var lastIndex;
$('#tt').datagrid({
    onClickRow:function(rowIndex){
        if (lastIndex != rowIndex){
            $('#tt').datagrid('endEdit', lastIndex);
            $('#tt').datagrid('beginEdit', rowIndex);
            setEditing(rowIndex);
        }
        lastIndex = rowIndex;
    }
});
```

为了在一些列之间创建运算关系，我们应该得到当前的 editors，并绑定一些事件到它们上面。

```
function setEditing(rowIndex){
    var editors = $('#tt').datagrid('getEditors', rowIndex);
    var priceEditor = editors[0];
    var amountEditor = editors[1];
    var costEditor = editors[2];
    priceEditor.target.bind('change', function(){
        calculate();
    });
    amountEditor.target.bind('change', function(){
        calculate();
    });
    function calculate(){
        var cost = priceEditor.target.val() * amountEditor.targ
        $(costEditor.target).numberbox('setValue',cost);
    }
}
```

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid15.zip

jQuery EasyUI 数据网格 - 合并单元格

数据网格（datagrid）经常需要合并一些单元格。本教程将向您展示如何在数据网格（datagrid）中合并单元格。

为了合并数据网格（datagrid）单元格，只需简单地调用 'mergeCells' 方法，并传入合并信息参数，告诉数据网格（datagrid）如何合并单元格。在所有合并的单元格中，除了第一个单元格，其它单元格在合并后被隐藏。

Merge Cells

	Product	Item ID	Price		Attribute	Status
			List Price	Unit Cost		
1	Koi	EST-1	16.5	10	Large	P
2	Dalmation	EST-10	18.5	12	Spotted Adult Female	P
3	Rattlesnake	EST-11	18.5	12	Venomless	P
4		EST-12	18.5	12	Rattleless	P
5	Iguana	EST-13	18.5	12	Green Adult	P
6	Manx	EST-14	58.5	12	Tailless	P
7		EST-15	23.5	12	With tail	P
8	Persian	EST-16	93.5	12	Adult Female	P
9		EST-17				

10 Page 1 of 3 Displaying 1 to 10 of 28 items

创建数据网格（DataGrid）

```
<table id="tt" title="Merge Cells" style="width:550px;height:25
        url="data/datagrid_data.json"
        singleSelect="true" iconCls="icon-save" rownumbers="tru
        idField="itemid" pagination="true">
    <thead frozen="true">
        <tr>
            <th field="productid" width="80" formatter="formatP
            <th field="itemid" width="100">Item ID</th>
        </tr>
    </thead>
    <thead>
        <tr>
            <th colspan="2">Price</th>
            <th rowspan="2" field="attr1" width="150">Attribute
            <th rowspan="2" field="status" width="60" align="ce
        </tr>
        <tr>
            <th field="listprice" width="80" align="right">List
            <th field="unitcost" width="80" align="right">Unit
        </tr>
    </thead>
</table>
```

合并单元格

当数据加载之后，我们合并数据网格（datagrid）中的一些单元格，所以放置下面的代码在 onLoadSuccess 回调函数中。

```
$('#tt').datagrid({
    onLoadSuccess:function(){
        var merges = [{
            index:2,
            rowspan:2
        },{
            index:5,
            rowspan:2
        },{
            index:7,
            rowspan:2
        }];
        for(var i=0; i<merges.length; i++)
            $('#tt').datagrid('mergeCells',{
                index:merges[i].index,
                field:'productid',
                rowspan:merges[i].rowspan
            });
    }
});
```

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid13.zip

jQuery EasyUI 数据网格 - 创建自定义视图

在不同的情况下，您可能需要为数据网格（datagrid）运用更灵活的布局。对于用户来说，卡片视图（Card View）是个不错的选择。这个工具可以在数据网格（datagrid）中迅速获取和显示数据。在数据网格（datagrid）的头部，您可以仅仅通过点击列的头部来排序数据。本教程将向您展示如何创建自定义卡片视图（Card View）。

DataGrid - CardView

Item ID | List Price | Unit Cost | Attribute | Status

Item ID: EST-1
List Price: 16.5
Unit Cost: 10
Attribute: Large
Status: P

Item ID: EST-10
List Price: 18.5
Unit Cost: 12
Attribute: Spotted Adult Female
Status: P

创建卡片视图（Card View）

从数据网格（datagrid）的默认视图继承，是个创建自定义视图的不错方法。我们将要创建一个卡片视图（Card View）来为每行显示一些信息。

```
var cardview = $.extend({}, $.fn.datagrid.defaults.view, {
    renderRow: function(target, fields, frozen, rowIndex, rowDa
        var cc = [];
        cc.push('<td colspan=' + fields.length + ' style="paddi
        if (!frozen){
            var aa = rowData.itemid.split('-');
            var img = 'shirt' + aa[1] + '.gif';
            cc.push('<img src="images/' + img + '" style="widtl
            cc.push('<div style="float:left;margin-left:20px;">
            for(var i=0; i<fields.length; i++){
                var copts = $(target).datagrid('getColumnOptior
                cc.push('<p><span class="c-label">' + copts.tit
            }
            cc.push('</div>');
        }
        cc.push('</td>');
        return cc.join('');
    }
});
```

创建数据网格（DataGrid）

现在我们使用视图创建数据网格（datagrid）。

```
<table id="tt" style="width:500px;height:400px"
        title="DataGrid - CardView" singleSelect="true" fitColu
        url="datagrid8_getdata.php" pagination="true" sortOrder
    <thead>
        <tr>
            <th field="itemid" width="80" sortable="true">Item
            <th field="listprice" width="80" sortable="true">Li
            <th field="unitcost" width="80" sortable="true">Uni
            <th field="attr1" width="150" sortable="true">Attri
            <th field="status" width="60" sortable="true">Statu
        </tr>
    </thead>
</table>
```

```
$('#tt').datagrid({
    view: cardview
});
```

请注意，我们设置 view 属性，且它的值为我们的卡片视图。

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid16.zip

jQuery EasyUI 数据网格 - 创建页脚摘要

在本教程中，我们将向您展示如何在数据网格（datagrid）页脚显示摘要信息行。

DataGrid

	Product Name	Unit Price
1	Chai	18
2	Chang	19
3	Aniseed Syrup	10
4	Chef Anton's Cajun Seasoning	22
5	Chef Anton's Gumbo Mix	21.35
6	Grandma's Boysenberry Spread	25
	Total	282.35
	Average	31.37

为了显示页脚行，您应该设置 showFooter 属性为 true，然后准备定义在数据网格（datagrid）数据中的页脚行。以下是示例数据：

```
{"total":1,"rows":[{"id":1,"name":"Chai","price":18.00}],"foote
```

创建数据网格（DataGrid）

```
<table id="tt" title="DataGrid" class="easyui-datagrid" style='
        url="data/datagrid17_data.json"
        fitColumns="true" rownumbers="true" showFooter="true">
    <thead>
        <tr>
            <th field="name" width="80">Product Name</th>
            <th field="price" width="40" align="right">Unit Pr:
        </tr>
    </thead>
</table>
```

页脚行和显示数据行一样，所以您可以在页脚显示不止一个摘要信息。

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid17.zip

jQuery EasyUI 数据网格 - 条件设置行背景颜色

本教程将向您展示如何根据一些条件改变数据网格（datagrid）组件的行样式。当 listprice 值大于 50 时，我们将为该行设置不同的颜色。

DataGrid

Item ID	Product ID	List Price	Unit Cost	Attribute	Status
EST-1	FI-SW-01	36.5	10	Large	P
EST-10	K9-DL-01	18.5	12	Spotted Adult Female	P
EST-11	RP-SN-01	28.5	12	Venomless	P
EST-12	RP-SN-01	26.5	12	Rattleless	P
EST-13	RP-LI-02	35.5	12	Green Adult	P
EST-14	**FL-DSH-01**	**158.5**	**12**	**Tailless**	**P**
EST-15	**FL-DSH-01**	**83.5**	**12**	**With tail**	**P**
EST-16	FL-DLH-02	63.5	12	Adult Female	P
EST-17	**FL-DLH-02**	**89.5**	**12**	**Adult Male**	**P**
EST-1	FI-SW-01	36.5	10	Large	P
EST-10	K9-DL-01	18.5	12	Spotted Adult Female	P
EST-11	RP-SN-01	28.5	12	Venomless	P

数据网格（datagrid）的 rowStyler 函数的设计目的是允许您自定义行样式。以下代码展示如何改变行样式：

```
<table id="tt" title="DataGrid" style="width:600px;height:250px
        url="data/datagrid_data.json"
        singleSelect="true" fitColumns="true">
    <thead>
        <tr>
            <th field="itemid" width="80">Item ID</th>
            <th field="productid" width="80">Product ID</th>
            <th field="listprice" width="80" align="right">List
            <th field="unitcost" width="80" align="right">Unit
            <th field="attr1" width="150">Attribute</th>
            <th field="status" width="60" align="center">Stauts
        </tr>
    </thead>
</table>
```

```
$('#tt').datagrid({
    rowStyler:function(index,row){
        if (row.listprice>50){
            return 'background-color:pink;color:blue;font-weigh
        }
    }
});
```

正如您所看到的，我们根据一些条件设置 background-color（背景色）为 pink（粉红色），设置文本颜色为 blue（蓝色）。

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid18.zip

jQuery EasyUI 数据网格 - 创建属性网格

属性网格（property grid）带有一个内置的 expand（展开）/collapse（合并） 按钮，可以简单地为行分组。您可以简单地创建一个可编辑属性的分层（hierarchical）列表。

Name	Value
⊟ ID Settings	
Name	Bill Smith
Address	
Age	40
Birthday	01/02/2012
SSN	123-456-7890
⊟ Marketing Settings	
Email	bill@gmail.com
FrequentBuyer	false

设置 HTML

```
<table id="tt" class="easyui-propertygrid" style="width:300px"
        url="propertygrid_data.json"
        showGroup="true" scrollbarSize="0"
></table>
```

准备 json 数据

```
[
    {"name":"Name","value":"Bill Smith","group":"ID Settings","
    {"name":"Address","value":"","group":"ID Settings","editor"
    {"name":"Age","value":"40","group":"ID Settings","editor":'
    {"name":"Birthday","value":"01/02/2012","group":"ID Settin
    {"name":"SSN","value":"123-456-7890","group":"ID Settings",
    {"name":"Email","value":"bill@gmail.com","group":"Marketin
        "type":"validatebox",
        "options":{
            "validType":"email"
        }
    }},
    {"name":"FrequentBuyer","value":"false","group":"Marketing
        "type":"checkbox",
        "options":{
            "on":true,
            "off":false
        }
    }}
]
```

正如您所看到的，属性网格（property grid）的创建不需要任何的 javascript 代码。您可以简单地继承扩展 editor 类型。

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid20.zip

jQuery EasyUI 数据网格 - 扩展行显示细节

数据网格（datagrid）可以改变它的视图（view）来显示不同的效果。使用详细视图，数据网格（datagrid）可以在数据行的左边显示展开按钮（"+" 或者 "-"）。用户可以展开行来显示附加的详细信息。

DataGrid - Expand Row

	Item ID	Product ID	List Price	Unit Cost	Status
+	EST-1	FI-SW-01	16.5	10	P
−	EST-10	K9-DL-01	18.5	12	P
	Item ID: EST-10 **List Price:** 18.5 **Attribute:** Spotted Adult Female	**Product ID:** K9-DL-01 **Unit Cost:** 12			
−	EST-11	RP-SN-01	18.5	12	P
	Item ID: EST-11 **List Price:** 18.5 **Attribute:** Venomless	**Product ID:** RP-SN-01 **Unit Cost:** 12			

步骤 1：创建数据网格（DataGrid）

```
<table id="dg" style="width:500px;height:250px"
        url="datagrid8_getdata.php"
        pagination="true" sortName="itemid" sortOrder="desc"
        title="DataGrid - Expand Row"
        singleSelect="true" fitColumns="true">
    <thead>
        <tr>
            <th field="itemid" width="60">Item ID</th>
            <th field="productid" width="80">Product ID</th>
            <th field="listprice" align="right" width="70">List
            <th field="unitcost" align="right" width="70">Unit
            <th field="status" width="50" align="center">Status
        </tr>
    </thead>
</table>
```

步骤 2：为数据网格（DataGrid）设置详细视图

为了使用详细视图，请记得在页面头部引用视图脚本文件。

```
<script type="text/javascript" src="http://www.w3cschool.cc/try/jea
```

```
$('#dg').datagrid({
    view: detailview,
    detailFormatter:function(index,row){
        return '<div class="ddv" style="padding:5px 0"></div>';
    },
    onExpandRow: function(index,row){
        var ddv = $(this).datagrid('getRowDetail',index).find('div
        ddv.panel({
            border:false,
            cache:false,
            href:'datagrid21_getdetail.php?itemid='+row.itemid,
            onLoad:function(){
                $('#dg').datagrid('fixDetailRowHeight',index);
            }
        });
        $('#dg').datagrid('fixDetailRowHeight',index);
    }
});
```

我们定义 'detailFormatter' 函数，告诉数据网格（datagrid）如何渲染详细视图。在这种情况下,我们返回一个简单的 '<div>' 元素，它将充当详细内容的容器。请注意，详细信息为空。当用户点击展开按钮（'+'）时，onExpandRow 事件将被触发。所以我们可以写一些代码来加载 ajax 详细内容。最后我们调用 'fixDetailRowHeight' 方法来固定当详细内容加载时的行高度。

步骤 3：服务器端代码

datagrid21_getdetail.php

```
&lt;?php
    include_once 'conn.php';

    $itemid = mysql_real_escape_string($_REQUEST['itemid']);

    $rs = mysql_query("select * from item where itemid='$itemid'");
    $item = mysql_fetch_array($rs);
?&gt;

<table class="dv-table" border="0" style="width:100%;">
    <tr>
        <td rowspan="3" style="width:60px">
            &lt;?php
                $aa = explode('-',$itemid);
                $serno = $aa[1];
                $img = "images/shirt$serno.gif";
                echo "<img src=\"$img\" style=\"width:60px;margin-r
            ?&gt;
        </td>
        <td class="dv-label">Item ID: </td>
        <td>&lt;?php echo $item['itemid'];?&gt;</td>
        <td class="dv-label">Product ID:</td>
        <td>&lt;?php echo $item['productid'];?&gt;</td>
    </tr>
    <tr>
        <td class="dv-label">List Price: </td>
        <td>&lt;?php echo $item['listprice'];?&gt;</td>
        <td class="dv-label">Unit Cost:</td>
        <td>&lt;?php echo $item['unitcost'];?&gt;</td>
    </tr>
    <tr>
        <td class="dv-label">Attribute: </td>
        <td colspan="3">&lt;?php echo $item['attr1'];?&gt;</td>
    </tr>
</table>
```

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid21.zip

jQuery EasyUI 数据网格 - 创建子网格

使用数据网格（datagrid）的详细视图，用户可以展开一行来显示附加的详细信息。 任何内容都可以加载作为行详细，子网格也可以动态加载。 本教程将向您展示如何在主网格上创建一个子网格。

DataGrid - SubGrid

	Item ID	Product ID	List Price	Unit Cost	Attribute	Status
+	EST-12	RP-SN-01	18.50	12.00	Rattleless	P
−	EST-13	RP-LI-02	18.50	12.00	Green Adult	P

	Order ID	Quantity	Unit Price
1	1002	1	18.50
2	1003	2	18.50
3	1010	1	18.50
4	1015	1	18.50

	Item ID	Product ID	List Price	Unit Cost	Attribute	Status
+	EST-16	FL-DLH-02	93.50	12.00	Adult Female	P
+	EST-18	AV-CB-01	193.50	92.00	Adult Male	P

步骤 1：创建主网格

```
<table id="dg" style="width:700px;height:250px"
        url="datagrid22_getdata.php"
        title="DataGrid - SubGrid"
        singleSelect="true" fitColumns="true">
    <thead>
        <tr>
            <th field="itemid" width="80">Item ID</th>
            <th field="productid" width="100">Product ID</th>
            <th field="listprice" align="right" width="80">List Pr
            <th field="unitcost" align="right" width="80">Unit Cost
            <th field="attr1" width="220">Attribute</th>
            <th field="status" width="60" align="center">Status</th
        </tr>
    </thead>
</table>
```

步骤 2：设置详细视图来显示子网格

为了使用详细视图，请记得在页面头部引用视图脚本文件。

```
<script type="text/javascript" src="http://www.w3cschool.cc/try/jea
```

```
$('#dg').datagrid({
    view: detailview,
    detailFormatter:function(index,row){
        return '<div style="padding:2px"><table class="ddv"></table
    },
    onExpandRow: function(index,row){
        var ddv = $(this).datagrid('getRowDetail',index).find('tabl
        ddv.datagrid({
            url:'datagrid22_getdetail.php?itemid='+row.itemid,
            fitColumns:true,
            singleSelect:true,
            rownumbers:true,
            loadMsg:'',
            height:'auto',
            columns:[[
                {field:'orderid',title:'Order ID',width:100},
                {field:'quantity',title:'Quantity',width:100},
                {field:'unitprice',title:'Unit Price',width:100}
            ]],
            onResize:function(){
                $('#dg').datagrid('fixDetailRowHeight',index);
            },
            onLoadSuccess:function(){
                setTimeout(function(){
                    $('#dg').datagrid('fixDetailRowHeight',index);
                },0);
            }
        });
        $('#dg').datagrid('fixDetailRowHeight',index);
    }
});
```

当用户点击展开按钮（'+'）时，'onExpandRow' 事件将被触发。 我们创建一个新的带有三列的子网格。 当子网格数据加载成功时或者改变尺寸大小时，请记得对主网格调用 'fixDetailRowHeight' 方法。

步骤 3：服务器端代码

datagrid22_getdata.php

```
$result = array();

include 'conn.php';

$rs = mysql_query("select * from item where itemid in (select item

$items = array();
while($row = mysql_fetch_object($rs)){
    array_push($items, $row);
}

echo json_encode($items);
```

datagrid22_getdetail.php

```
include 'conn.php';

$itemid = mysql_real_escape_string($_REQUEST['itemid']);

$rs = mysql_query("select * from lineitem where itemid='$itemid'")
$items = array();
while($row = mysql_fetch_object($rs)){
    array_push($items, $row);
}
echo json_encode($items);
```

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid22.zip

jQuery EasyUI 数据网格 - 使用虚拟滚动视图显示海量数据

数据网格（datagrid）的虚拟滚动特性可以用来显示大数量的记录而不需要分页。当滚动垂直滚动条时，数据网格（datagrid）执行 ajax 请求来加载和刷新现有的记录。 整个刷新的行为过程是平稳的没有闪烁。 在本教程中，我们将创建一个数据网格（datagrid），并运用虚拟滚动特性从服务器加载数据。

DataGrid - VirtualScrollView

	Inv No	Date	Name	Amount	Price	Cost	Note
6279	INV6279	2012-10-05	Name29	94	113.01	10622.94	Note29
6280	INV6280	2012-10-06	Name30	97	196.41	19051.77	Note30
6281	INV6281	2012-10-07	Name31	81	111.97	9069.57	Note31
6282	INV6282	2012-10-08	Name32	81	181.21	14678.01	Note32
6283	INV6283	2012-10-09	Name33	68	159.96	10877.28	Note33
6284	INV6284	2012-10-10	Name34	87	113.47	9871.89	Note34
6285	INV6285	2012-10-11	Name35	92	191.35	17604.20	Note35
6286	INV6286	2012-10-12	Name36	88	102.56	9025.28	Note36
6287	INV6287	2012-10-13	Name37	63	190.51	12002.13	Note37
6288	INV6288	2012-10-14	Name38	83	107.88	8954.04	Note38

创建数据网格（DataGrid）

为数据网格（datagrid）运用虚拟滚动特性，'view' 属性应该设置为 'scrollview'。用户应该从数据网格（datagrid）扩展下载 scrollview，并在页面头部引用 scrollview 文件。

```
<script type="text/javascript" src="http://www.w3cschool.cc/try/jea
```

```
<table id="tt" class="easyui-datagrid" style="width:700px;height:30
        title="DataGrid - VirtualScrollView"
        data-options="view:scrollview,rownumbers:true,singleSelect
            url:'datagrid27_getdata.php',autoRowHeight:false,pageSi
    <thead>
        <tr>
            <th field="inv" width="80">Inv No</th>
            <th field="date" width="100">Date</th>
            <th field="name" width="80">Name</th>
            <th field="amount" width="80" align="right">Amount</th
            <th field="price" width="80" align="right">Price</th>
            <th field="cost" width="100" align="right">Cost</th>
            <th field="note" width="110">Note</th>
        </tr>
    </thead>
</table>
```

请注意，这里我们不需要使用 pagination 属性，但 pageSize 属性是必需的，这样执行 ajax 请求时，数据网格（datagrid）将从服务器获取指定数量的记录。

服务器端代码

datagrid27_getdata.php

```
$page = isset($_POST['page']) ? intval($_POST['page']) : 1;
$rows = isset($_POST['rows']) ? intval($_POST['rows']) : 50;

$items = array();
date_default_timezone_set('UTC');
for($i=1; $i<=$rows; $i++){
    $index = $i+($page-1)*$rows;
    $amount = rand(50,100);
    $price = rand(10000,20000)/100;
    $items[] = array(
        'inv' => sprintf("INV%04d",$index),
        'date' => date('Y-m-d',time()+24*3600*$i),
        'name' => 'Name' . $index,
        'note' => 'Note' . $index,
        'amount' => $amount,
        'price' => sprintf('%01.2f',$price),
        'cost' => sprintf('%01.2f',$amount*$price)
    );
}
$result = array();
$result['total'] = 8000;
$result['rows'] = $items;
echo json_encode($result);
```

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid27.zip

jQuery EasyUI 数据网格 - 添加分页组件

本实例演示如何从服务器端加载数据，如何添加分页组件（pagination）到数据网格（datagrid）。

Load Data

	Item ID	Product ID	List Price	Unit Cost	Attribute	Stauts
11	EST-19	AV-SB-02	15.50	2.00	Adult Male	P
12	EST-2	FI-SW-01	16.50	10.00	Small	P
13	EST-20	FI-FW-02	5.50	2.00	Adult Male	P
14	EST-21	FI-FW-02	5.29	1.00	Adult Female	P
15	EST-22	K9-RT-02	135.50	100.00	Adult Male	P
16	EST-23	K9-RT-02	145.49	100.00	Adult Female	P
17	EST-24	K9-RT-02	255.50	92.00	Adult Male	P
18	EST-25	K9-RT-02	325.29	90.00	Adult Female	P

10 Page 2 of 3 Displaying 11 to 20 of 28 items

创建数据网格（DataGrid）

为了从远程服务器端加载数据，您应该设置 'url' 属性，在您的服务器端应该返回 JSON 格式数据。请看数据网格（datagrid）文档得到更多关于它的数据格式信息。

```
<table id="tt" class="easyui-datagrid" style="width:600px;heigh
        url="datagrid2_getdata.php"
        title="Load Data" iconCls="icon-save"
        rownumbers="true" pagination="true">
    <thead>
        <tr>
            <th field="itemid" width="80">Item ID</th>
            <th field="productid" width="80">Product ID</th>
            <th field="listprice" width="80" align="right">List
            <th field="unitcost" width="80" align="right">Unit
            <th field="attr1" width="150">Attribute</th>
            <th field="status" width="60" align="center">Stauts
        </tr>
    </thead>
</table>
```

我们定义数据网格（datagrid）列，并设置 'pagination' 属性为 true，它将在数据网格（datagrid）的底部生成一个分页（pagination）工具栏。pagination将发送两个参数到服务器：

- page：页码，起始值 1。

- rows：每页显示行。

服务器端代码

```
$page = isset($_POST['page']) ? intval($_POST['page']) : 1;
$rows = isset($_POST['rows']) ? intval($_POST['rows']) : 10;
// ...
$rs = mysql_query("select count(*) from item");
$row = mysql_fetch_row($rs);
$result["total"] = $row[0];

$rs = mysql_query("select * from item limit $offset,$rows");

$items = array();
while($row = mysql_fetch_object($rs)){
    array_push($items, $row);
}
$result["rows"] = $items;

echo json_encode($result);
```

下载 jQuery EasyUI 实例

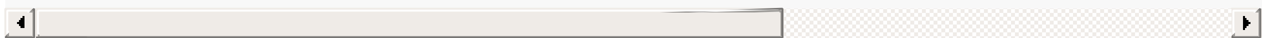
jeasyui-datagrid-datagrid2.zip

jQuery EasyUI 窗口

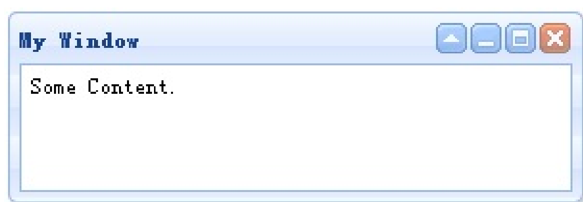
jQuery EasyUI 窗口 - 创建简单窗口

创建一个窗口（window）非常简单，我们创建一个 DIV 标记：

```
<div id="win" class="easyui-window" title="My Window" style="width
    Some Content.
</div>
```

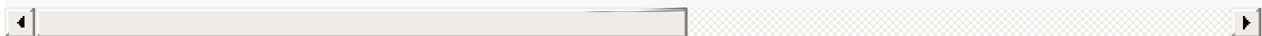


现在运行测试页面，您会看见一个窗口（window）显示在您的屏幕上。我们不需要写任何的 javascript 代码。



如果您希望创建一个隐藏的窗口（window），记得设置 'closed' 属性为 'true' 值，您可以调用 'open' 方法来打开窗口（window）：

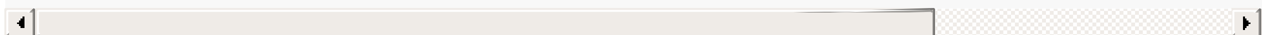
```
<div id="win" class="easyui-window" title="My Window" closed="true'
    Some Content.
</div>
```



```
$('#win').window('open');
```

作为最后的实例演示，我们创建一个登录窗口（window）：

```
<div id="win" class="easyui-window" title="Login" style="width:300p
    <form style="padding:10px 20px 10px 40px;">
        <p>Name: <input type="text"></p>
        <p>Pass: <input type="password"></p>
        <div style="padding:5px;text-align:center;">
            <a href="#" class="easyui-linkbutton" icon="icon-ok">Ok
            <a href="#" class="easyui-linkbutton" icon="icon-cancel
        </div>
    </form>
</div>
```





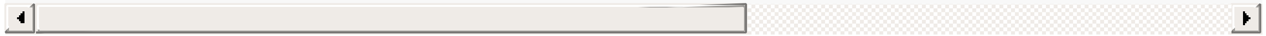
下载 jQuery EasyUI 实例

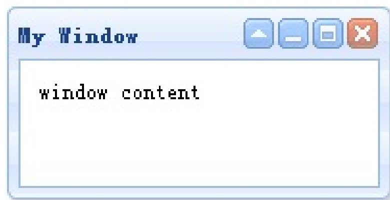
jeasyui-win-win1.zip

jQuery EasyUI 窗口 - 自定义窗口工具栏

默认情况下，窗口（window）有四个工具：collapsible、minimizable、maximizable 和 closable。比如我们定义以下窗口（window）：

```
<div id="win" class="easyui-window" title="My Window" style="pa
    window content
</div>
```





如需自定义工具，设置该工具为 true 或者 false。比如我们希望定义一个窗口（window），仅仅拥有一个可关闭的工具。您应该设置任何其他工具为 false。我们可以在标记中或者通过 jQuery 代码定义 tools 属性。现在我们使用 jQuery 代码来定义窗口（window）：

```
$('#win').window({
    collapsible:false,
    minimizable:false,
    maximizable:false
});
```



如果我们希望添加自定义的工具到窗口（window），我们可以使用 tools 属性。作为实例演示，我们添加两个工具到窗口（window）：

```
$('#win').window({
    collapsible:false,
    minimizable:false,
    maximizable:false,
    tools:[{
        iconCls:'icon-add',
        handler:function(){
            alert('add');
        }
    },{
        iconCls:'icon-remove',
        handler:function(){
            alert('remove');
        }
    }]
});
```



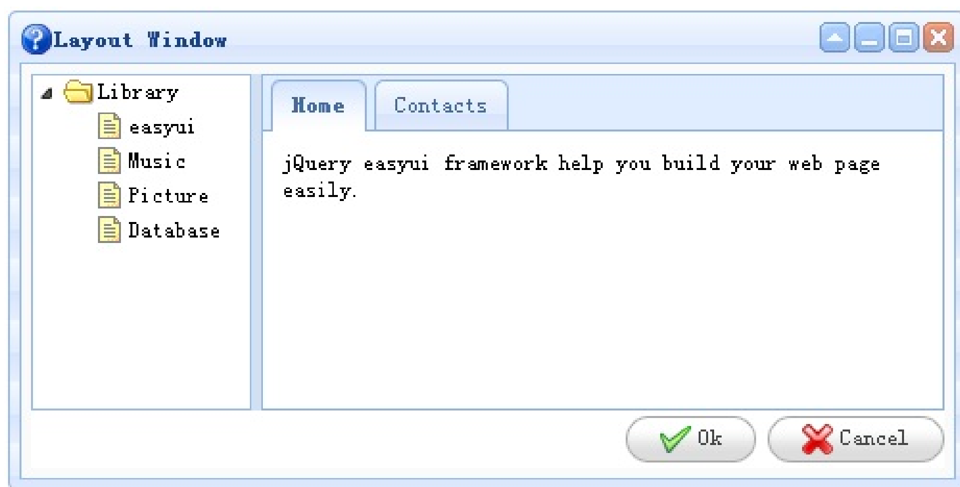
下载 jQuery EasyUI 实例

jeasyui-win-win2.zip

jQuery EasyUI 窗口 - 窗口与布局

Layout 组件可以内嵌在窗口（window）中。我们可以创建一个复杂的布局窗口，甚至不需要写任何的 js 代码。jquery-easyui 框架帮我们在后台做渲染和调整尺寸。

作为一个实例，我们创建一个窗口（window），它包含两个部分，一个放置在左边一个放置在右边。在窗口（window）的左边我们创建一个树形菜单（tree），在窗口（window）的右边我们创建一个 tabs 容器。



```
    <div class="easyui-window" title="Layout Window" icon="icon-hel
        <div class="easyui-layout" fit="true">
            <div region="west" split="true" style="width:120px;">
                <ul class="easyui-tree">
                    <li>
                        <span>Library</span>
                        <ul>
                            <li><span>easyui</span></li>
                            <li><span>Music</span></li>
                            <li><span>Picture</span></li>
                            <li><span>Database</span></li>
                        </ul>
                    </li>
                </ul>
            </div>
            <div region="center" border="false" border="false">
                <div class="easyui-tabs" fit="true">
                    <div title="Home" style="padding:10px;">
                        jQuery easyui framework help you build your
                    </div>
                    <div title="Contacts">
                        No contact data.
                    </div>
                </div>
            </div>
            <div region="south" border="false" style="text-align:ri
                <a class="easyui-linkbutton" icon="icon-ok" href="j
                <a class="easyui-linkbutton" icon="icon-cancel" hre
            </div>
        </div>
    </div>
```

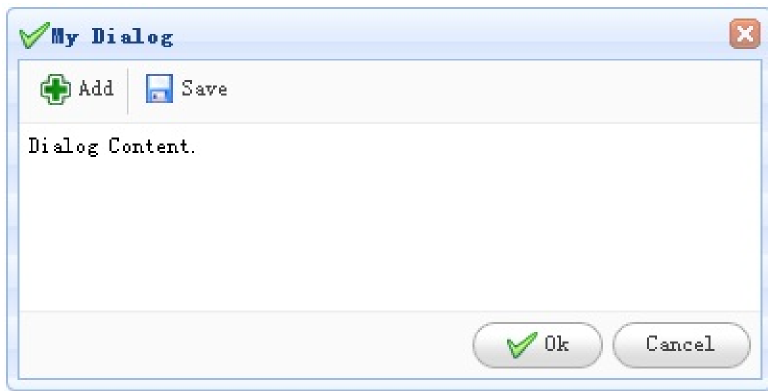
请看上面的代码，我们仅仅使用了 HTML 标记，一个复杂的布局窗口（window）将显示。这就是 jquery-easyui 框架，简单而强大。

下载 jQuery EasyUI 实例

jeasyui-win-win3.zip

jQuery EasyUI 窗口 - 创建对话框

对话框（Dialog）是一个特殊的窗口（window），可以包含在顶部的工具栏和在底部的按钮。默认情况下，对话框（Dialog）不能改变大小，但是用户可以设置 resizable 属性为 true，使其可以改变大小。



创建对话框（Dialog）

对话框（Dialog）非常简单，可以从 DIV 标记创建，如下所示：

```
<div id="dd" class="easyui-dialog" style="padding:5px;width:40(
        title="My Dialog" iconCls="icon-ok"
        toolbar="#dlg-toolbar" buttons="#dlg-buttons">
    Dialog Content.
</div>
```

准备工具栏（Toolbar）和按钮（Button）

```
<div id="dlg-toolbar">
    <a href="#" class="easyui-linkbutton" iconCls="icon-add" pl
    <a href="#" class="easyui-linkbutton" iconCls="icon-save" p
</div>
<div id="dlg-buttons">
    <a href="#" class="easyui-linkbutton" iconCls="icon-ok" onc
    <a href="#" class="easyui-linkbutton" iconCls="icon-cancel'
</div>
```

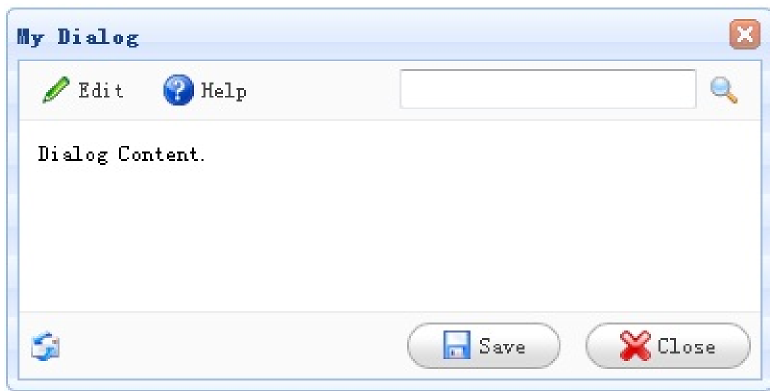
上面的代码我们创建了一个带有工具栏（toolbar）和按钮（button）的对话框（dialog）。这是对话框（dialog）、工具栏（toolbar）、内容（content）和按钮（buttons）的标准配置。

下载 jQuery EasyUI 实例

jeasyui-win-dlg1.zip

jQuery EasyUI 窗口 - 自定义带有工具条和按钮的对话框

您可以创建一个带有工具栏（toolbar）和按钮（button）的对话框（dialog），可以从 HTML 标记创建。这个教程描述如何添加工具栏（toolbar）和按钮（button）到对话框（dialog），没有任何的 javascript 代码。



创建对话框（Dialog）

```
    <div id="dd" class="easyui-dialog" title="My Dialog" style="wi
            toolbar="#dlg-toolbar" buttons="#dlg-buttons">
        Dialog Content.
    </div>
```

创建工具栏（Toolbar）

```
    <div id="dlg-toolbar">
        <table cellpadding="0" cellspacing="0" style="width:100%">
            <tr>
                <td>
                    <a href="#" class="easyui-linkbutton" iconCls='
                    <a href="#" class="easyui-linkbutton" iconCls='
                </td>
                <td style="text-align:right">
                    <input></input><a href="#" class="easyui-linkbu
                </td>
            </tr>
        </table>
    </div>
```

创建按钮（Buttons）

```
    <div id="dlg-buttons">
        <table cellpadding="0" cellspacing="0" style="width:100%">
            <tr>
                <td>
                    <img src="email.gif"/>
                </td>
                <td style="text-align:right">
                    <a href="#" class="easyui-linkbutton" iconCls='
                    <a href="#" class="easyui-linkbutton" iconCls='
                </td>
            </tr>
        </table>
    </div>
```

请注意，对话框（dialog）的工具栏（toolbar）和按钮（buttons）属性也可以通过 string 值指定，它将充当作为一个选择器去选择一个适当的 DIV 元素，并追加到工具栏（toolbar）或者按钮（buttons）的位置。

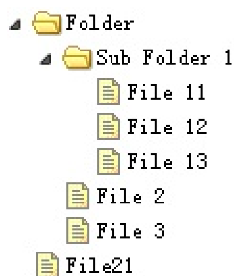
下载 jQuery EasyUI 实例

jeasyui-win-dlg2.zip

jQuery EasyUI 树形菜单

jQuery EasyUI 树形菜单 - 使用标记创建树形菜单

一个树形菜单（Tree）可以从标记创建。easyui 树形菜单（Tree）也可以定义在 <ul> 元素中。无序列表的 <ul> 元素提供一个基础的树（Tree）结构。每一个 <li> 元素将产生一个树节点，子 <ul> 元素将产生一个父树节点。



创建树形菜单（Tree）

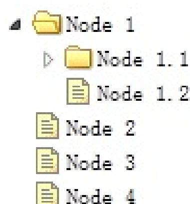
```
<ul class="easyui-tree">
    <li>
        <span>Folder</span>
        <ul>
            <li>
                <span>Sub Folder 1</span>
                <ul>
                    <li><span>File 11</span></li>
                    <li><span>File 12</span></li>
                    <li><span>File 13</span></li>
                </ul>
            </li>
            <li><span>File 2</span></li>
            <li><span>File 3</span></li>
        </ul>
    </li>
    <li><span>File21</span></li>
</ul>
```

下载 jQuery EasyUI 实例

jeasyui-tree-tree1.zip

jQuery EasyUI 树形菜单 - 创建异步树形菜单

为了创建异步的树形菜单（Tree），每一个树节点必须要有一个 'id' 属性，这个将提交回服务器去检索子节点数据。



创建树形菜单（Tree）

```
<ul id="tt" class="easyui-tree"
        url="tree2_getdata.php">
</ul>
```

服务器端代码

```
$id = isset($_POST['id']) ? intval($_POST['id']) : 0;

include 'conn.php';

$result = array();
$rs = mysql_query("select * from nodes where parentId=$id");
while($row = mysql_fetch_array($rs)){
    $node = array();
    $node['id'] = $row['id'];
    $node['text'] = $row['name'];
    $node['state'] = has_child($row['id']) ? 'closed' : 'open',
    array_push($result,$node);
}

echo json_encode($result);

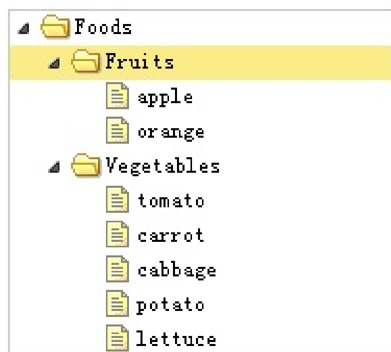
function has_child($id){
    $rs = mysql_query("select count(*) from nodes where parentI
    $row = mysql_fetch_array($rs);
    return $row[0] > 0 ? true : false;
}
```

下载 jQuery EasyUI 实例

jeasyui-tree-tree2.zip

jQuery EasyUI 树形菜单 - 树形菜单添加节点

本教程向您展示如何附加节点到树形菜单（Tree）。我们将创建一个包含水果和蔬菜节点的食品树，然后添加一些其他水果到已存在的水果节点。



创建食品树

首先，我们创建食品树，代码如下所示：

```
<div style="width:200px;height:auto;border:1px solid #ccc;">
    <ul id="tt" class="easyui-tree" url="tree_data.json"></ul>
</div>
```

请注意，树（Tree)组件是定义在 <ul> 标记中，树节点数据从 URL "tree_data.json" 加载。

得到父节点

然后我们通过点击节点选择水果节点，我们将添加一些其他的水果数据。执行 getSelected 方法得到处理节点：

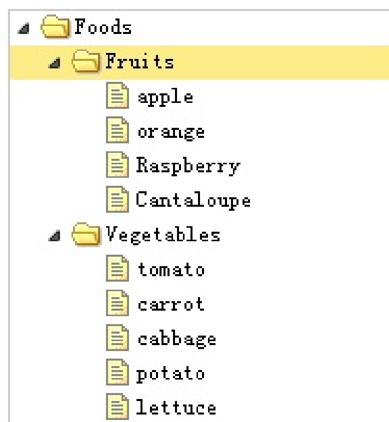
```
var node = $('#tt').tree('getSelected');
```

getSelected 方法的返回结果是一个 javascript 对象，它有一个 id、text、target 属性。target 属性是一个 DOM 对象，引用选中节点，它的 append 方法将用于附加子节点。

附加节点

```
var node = $('#tt').tree('getSelected');
if (node){
    var nodes = [{
        "id":13,
        "text":"Raspberry"
    },{
        "id":14,
        "text":"Cantaloupe"
    }];
    $('#tt').tree('append', {
        parent:node.target,
        data:nodes
    });
}
```

当添加一些水果，您将看见：



正如您所看到的，使用 easyui 的树（Tree）插件去附加节点不是那么的难。

下载 jQuery EasyUI 实例

jeasyui-tree-tree3.zip

jQuery EasyUI 树形菜单 - 创建带复选框的树形菜单

easyui 的树（Tree）插件允许您创建一个复选框树。如果您点击一个节点的复选框，这个点击的节点信息将向上和向下继承。例如：点击 'tomato' 节点的复选框，您将会看见 'Vegetables' 节点现在仅仅选中部分。



创建复选框树

```
<ul id="tt" class="easyui-tree"
        url="data/tree_data.json"
        checkbox="true">
</ul>
```

下载 jQuery EasyUI 实例

jeasyui-tree-tree4.zip

jQuery EasyUI 树形菜单 - 树形菜单拖放控制

当在一个应用中使用树（Tree）插件，拖拽（drag）和放置（drop）功能要求允许用户改变节点位置。启用拖拽（drag）和放置（drop）操作，所有您需要做的就是把树（Tree）插件的 'dnd' 属性设置为 true。



创建树形菜单（Tree）

```
$('#tt').tree({
    dnd: true,
    url: 'tree_data.json'
});
```

当在一个树节点上发生放置操作，'onDrop' 事件将被触发，您应该做一些或更多的操作，例如保存节点状态到远程服务器端，等等。

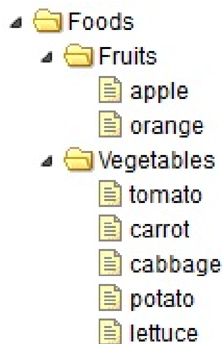
```
onDrop: function(targetNode, source, point){
    var targetId = $(target).tree('getNode', targetNode).id;
    $.ajax({
        url: '...',
        type: 'post',
        dataType: 'json',
        data: {
            id: source.id,
            targetId: targetId,
            point: point
        }
    });
}
```

下载 jQuery EasyUI 实例

jeasyui-tree-tree5.zip

jQuery EasyUI 树形菜单 - 树形菜单加载父/子节点

通常表示一个树节点的方式就是在每一个节点存储一个 parentid。 这个也被称为邻接列表模型。 直接加载这些数据到树形菜单（Tree）是不允许的。 但是我们可以在加载树形菜单之前，把它转换为标准标准的树形菜单（Tree）数据格式。 树（Tree）插件提供一个 'loadFilter' 选项函数，它可以实现这个功能。 它提供一个机会来改变任何一个进入数据。 本教程向您展示如何使用 'loadFilter' 函数加载父/子节点到树形菜单（Tree）。



父/子节点数据

```
[
{"id":1,"parendId":0,"name":"Foods"},
{"id":2,"parentId":1,"name":"Fruits"},
{"id":3,"parentId":1,"name":"Vegetables"},
{"id":4,"parentId":2,"name":"apple"},
{"id":5,"parentId":2,"name":"orange"},
{"id":6,"parentId":3,"name":"tomato"},
{"id":7,"parentId":3,"name":"carrot"},
{"id":8,"parentId":3,"name":"cabbage"},
{"id":9,"parentId":3,"name":"potato"},
{"id":10,"parentId":3,"name":"lettuce"}
]
```

使用 'loadFilter' 创建树形菜单（Tree）

```
$('#tt').tree({
    url: 'data/tree6_data.json',
    loadFilter: function(rows){
        return convert(rows);
    }
});
```

转换的实现

```
function convert(rows){
    function exists(rows, parentId){
        for(var i=0; i<rows.length; i++){
            if (rows[i].id == parentId) return true;
        }
        return false;
    }

    var nodes = [];
    // get the top level nodes
    for(var i=0; i<rows.length; i++){
        var row = rows[i];
        if (!exists(rows, row.parentId)){
            nodes.push({
                id:row.id,
                text:row.name
            });
        }
    }

    var toDo = [];
    for(var i=0; i<nodes.length; i++){
        toDo.push(nodes[i]);
    }
    while(toDo.length){
        var node = toDo.shift();    // the parent node
        // get the children nodes
        for(var i=0; i<rows.length; i++){
            var row = rows[i];
            if (row.parentId == node.id){
                var child = {id:row.id,text:row.name};
                if (node.children){
                    node.children.push(child);
                } else {
                    node.children = [child];
                }
                toDo.push(child);
            }
        }
    }
    return nodes;
}
```

下载 jQuery EasyUI 实例

jeasyui-tree-tree6.zip

jQuery EasyUI 树形菜单 - 创建基础树形网格

树形网格（TreeGrid）组件从数据网格（DataGrid）继承，但是允许在行之间存在父/子节点关系。许多属性继承至数据网格（DataGrid），可以用在树形网格（TreeGrid）中。为了使用树形网格（TreeGrid），用户必须定义 'treeField' 属性，指明哪个字段作为树节点。

本教程将向您展示如何使用树形网格（TreeGrid）组件设置一个文件夹浏览。

Folder Browser

	Name	Size	Modified Date
1	C		02/19/2010
2	Program Files	120 MB	03/20/2010
3	Java		01/13/2010
6	MySQL		01/13/2010
10	eclipse		01/20/2010
11	eclipse.exe	56 KB	05/19/2009
12	eclipse.ini	1 KB	04/20/2010
13	notice.html	7 KB	03/17/2005

创建树形网格（TreeGrid）

```
<table id="test" title="Folder Browser" class="easyui-treegrid'
        url="data/treegrid_data.json"
        rownumbers="true"
        idField="id" treeField="name">
    <thead>
        <tr>
            <th field="name" width="160">Name</th>
            <th field="size" width="60" align="right">Size</th
            <th field="date" width="100">Modified Date</th>
        </tr>
    </thead>
</table>
```

下载 jQuery EasyUI 实例

jeasyui-tree-treegrid1.zip

jQuery EasyUI 树形菜单 - 创建复杂树形网格

树形网格（TreeGrid）可以展示有限空间上带有多列和复杂数据电子表格。本教程将演示如何将表格数据排列在分割的网格和多行表头中，以便组织共同的数据。

Complex TreeGrid

	Region	2009			2010			
		2st qrt.	3st qrt.	4st qrt.	1st qrt.	2st qrt.	3st qrt.	4st qrt.
1	Wyoming							
2	Albin	1800	1903	2183	2133	1923	2018	1838
3	Canon	1800	1903	2183	2133	1923	2018	1838
4	Egbert	1800	1903	2183	2133	1923	2018	1838
5	Washington							
6	Bellingham							
	Total	12600	13321	15281	14931	13461	14126	12866

创建树形网格（TreeGrid）

```
<table title="Complex TreeGrid" class="easyui-treegrid" style='
        url="data/treegrid2_data.json"
        rownumbers="true" showFooter="true"
        idField="id" treeField="region">
    <thead frozen="true">
        <tr>
            <th field="region" width="150">Region</th>
        </tr>
    </thead>
    <thead>
        <tr>
            <th colspan="4">2009</th>
            <th colspan="4">2010</th>
        </tr>
        <tr>
            <th field="f1" width="50" align="right">1st qrt.</t
            <th field="f2" width="50" align="right">2st qrt.</t
            <th field="f3" width="50" align="right">3st qrt.</t
            <th field="f4" width="50" align="right">4st qrt.</t
            <th field="f5" width="50" align="right">1st qrt.</t
            <th field="f6" width="50" align="right">2st qrt.</t
            <th field="f7" width="50" align="right">3st qrt.</t
            <th field="f8" width="50" align="right">4st qrt.</t
        </tr>
    </thead>
</table>
```

正如您所看到的，树形网格（Treegrid）的使用和数据网格（Datagrid）一样。请使用 'frozen' 属性来定义冻结列，列的 'colspan' 和 'rowspan' 属性来定义多行表头。

下载 jQuery EasyUI 实例

jeasyui-tree-treegrid2.zip

jQuery EasyUI 树形菜单 - 树形网格动态加载

动态加载树形网格有助于从服务器上加载部分的行数据，避免加载大型数据的长时间等待。本教程将向您展示如何创建带有动态加载特性的树形网格（TreeGrid）。

Products

	Name	Quantity	Price	Total
1	Computers			
2	Printers			
3	Epson WorkForce 845	69	$121.29	$8369.01
4	Canon PIXMA MG5320	12	$110.12	$1321.44
5	HP Deskjet 1000 Printer	63	$33.95	$2138.85
6	Firewall			
7	Cisco RV110W-A-NA-K9	103	$79.95	$8234.85
8	ZyXEL ZyWALL USG50	34	$209.99	$7139.66
9	NETGEAR FVS318	67	$89.99	$6029.33
10	Keyboard			

创建树形网格（TreeGrid）

```
    <table title="Products" class="easyui-treegrid" style="width:70
            url="treegrid3_getdata.php"
            rownumbers="true"
            idField="id" treeField="name">
        <thead>
            <tr>
                <th field="name" width="250">Name</th>
                <th field="quantity" width="100" align="right">Quan
                <th field="price" width="150" align="right" formatt
                <th field="total" width="150" align="right" formatt
            </tr>
        </thead>
    </table>
```

服务器端代码

treegrid3_getdata.php

```
$id = isset($_POST['id']) ? intval($_POST['id']) : 0;

include 'conn.php';
$result = array();
$rs = mysql_query("select * from products where parentId=$id");
while($row = mysql_fetch_array($rs)){
    $row['state'] = has_child($row['id']) ? 'closed' : 'open';
    $row['total'] = $row['price']*$row['quantity'];
    array_push($result, $row);
}

echo json_encode($result);

function has_child($id){
    $rs = mysql_query("select count(*) from products where parentIc
    $row = mysql_fetch_array($rs);
    return $row[0] > 0 ? true : false;
}
```

下载 jQuery EasyUI 实例

jeasyui-tree-treegrid3.zip

jQuery EasyUI 树形菜单 - 树形网格添加分页

本教程展示如何向带有动态加载特性的树形网格（TreeGrid）添加分页。

Products

	Name	Quantity	Price	Total
1	Computers			
2	Electronics			
3	Digital Cameras			
4	Nikon COOLPIX L26 16.1 MP	12	$74.97	$899.64
5	Canon PowerShot A1300	143	$109.99	$15728.57
6	Canon PowerShot A2300	32	$91.52	$2928.64
7	DVD			
8	Verbatim 95101 4.7 GB	35	$11.81	$413.35
9	Brave	78	$14.99	$1169.22

2 Page 1 of 2 Displaying 1 to 2 of 3 items

创建树形网格（TreeGrid）

启用树形网格（TreeGrid）的分页特性，必须添加 'pagination:true' 属性，这样页面加载时就会向服务器发送 'page' 和 'rows' 参数。

```
<table title="Products" class="easyui-treegrid" style="width:70
        data-options="
            url: 'treegrid4_getdata.php',
            rownumbers: true,
            pagination: true,
            pageSize: 2,
            pageList: [2,10,20],
            idField: 'id',
            treeField: 'name',
            onBeforeLoad: function(row,param){
                if (!row) {    // load top level rows
                    param.id = 0;    // set id=0, indicate to l
                }
            }
        ">
    <thead>
        <tr>
            <th field="name" width="250">Name</th>
            <th field="quantity" width="100" align="right">Quan
            <th field="price" width="150" align="right" formatt
            <th field="total" width="150" align="right" formatt
        </tr>
    </thead>
</table>
```

服务器端代码

treegrid4_getdata.php

```
$page = isset($_POST['page']) ? intval($_POST['page']) : 1;
$rows = isset($_POST['rows']) ? intval($_POST['rows']) : 10;
$offset = ($page-1)*$rows;

$id = isset($_POST['id']) ? intval($_POST['id']) : 0;

include 'conn.php';

$result = array();
if ($id == 0){
    $rs = mysql_query("select count(*) from products where parentId
    $row = mysql_fetch_row($rs);
    $result["total"] = $row[0];

    $rs = mysql_query("select * from products where parentId=0 limi
    $items = array();
    while($row = mysql_fetch_array($rs)){
        $row['state'] = has_child($row['id']) ? 'closed' : 'open';
        array_push($items, $row);
    }
    $result["rows"] = $items;
} else {
    $rs = mysql_query("select * from products where parentId=$id");
    while($row = mysql_fetch_array($rs)){
        $row['state'] = has_child($row['id']) ? 'closed' : 'open';
        $row['total'] = $row['price']*$row['quantity'];
        array_push($result, $row);
    }
}

echo json_encode($result);

function has_child($id){
    $rs = mysql_query("select count(*) from products where parentId
    $row = mysql_fetch_array($rs);
    return $row[0] > 0 ? true : false;
}
```

发送到服务器的参数包括：

- page：要加载的当前页面。
- rows：页面尺寸大小。
- id：父行的 id 值，从服务器返回的行将被添加。

当展开一个行节点时，'id' 值是大于 0 的。 当改变页码时，'id' 值应该被设置为 0 来放置加载子行。

下载 jQuery EasyUI 实例

jeasyui-tree-treegrid4.zip

jQuery EasyUI 树形菜单 - 树形网格惰性加载节点

有时我们已经得到充分的分层树形网格（TreeGrid）的数据。 我们还想让树形网格（TreeGrid）按层次惰性加载节点。 首先，只加载顶层节点。 然后点击节点的展开图标来加载它的子节点。 本教程展示如何创建带有惰性加载特性的树形网格（TreeGrid）。

Lazy Loading

	Name	Size	Modified Date
1	C		02/19/2010
2	Program Files	120 MB	03/20/2010
3	Java		01/13/2010
4	MySQL		01/13/2010
5	eclipse		01/20/2010

创建树形网格（TreeGrid）

```
    <table id="test" title="Folder Browser" class="easyui-treegrid"
            data-options="
                url: 'data/treegrid_data.json',
                method: 'get',
                rownumbers: true,
                idField: 'id',
                treeField: 'name',
                loadFilter: myLoadFilter
            ">
        <thead>
            <tr>
                <th field="name" width="220">Name</th>
                <th field="size" width="100" align="right">Size</th>
                <th field="date" width="150">Modified Date</th>
            </tr>
        </thead>
    </table>
```

为了放置加载子节点，我们需要为每个节点重命名 'children' 属性。 正如下面的代码所示，'children' 属性重命名为 'children1'。 当展开一个节点时，我们调用 'append' 方法来加载它的子节点数据。

'loadFilter' 代码

```
function myLoadFilter(data,parentId){
    function setData(){
        var todo = [];
        for(var i=0; i<data.length; i++){
            todo.push(data[i]);
        }
        while(todo.length){
            var node = todo.shift();
            if (node.children){
                node.state = 'closed';
                node.children1 = node.children;
                node.children = undefined;
                todo = todo.concat(node.children1);
            }
        }
    }

    setData(data);
    var tg = $(this);
    var opts = tg.treegrid('options');
    opts.onBeforeExpand = function(row){
        if (row.children1){
            tg.treegrid('append',{
                parent: row[opts.idField],
                data: row.children1
            });
            row.children1 = undefined;
            tg.treegrid('expand', row[opts.idField]);
        }
        return row.children1 == undefined;
    };
    return data;
}
```

下载 jQuery EasyUI 实例

jeasyui-tree-treegrid5.zip

jQuery EasyUI 表单

jQuery EasyUI 表单 - 创建异步提交表单

本教程向您展示如何通过 easyui 提交一个表单（Form）。我们创建一个带有 name、email 和 phone 字段的表单。通过使用 easyui 表单（form）插件来改变表单（form）为 ajax 表单（form）。表单（form）提交所有字段到后台服务器，服务器处理和发送一些数据返回到前端页面。我们接收返回数据，并将它显示出来。

创建表单（Form）

```
<div style="padding:3px 2px;border-bottom:1px solid #ccc">Ajax
<form id="ff" action="form1_proc.php" method="post">
    <table>
        <tr>
            <td>Name:</td>
            <td><input name="name" type="text"></input></td>
        </tr>
        <tr>
            <td>Email:</td>
            <td><input name="email" type="text"></input></td>
        </tr>
        <tr>
            <td>Phone:</td>
            <td><input name="phone" type="text"></input></td>
        </tr>
        <tr>
            <td></td>
            <td><input type="submit" value="Submit"></input></t
        </tr>
    </table>
</form>
```

改变为 Ajax 表单

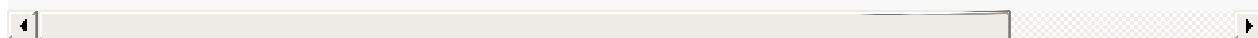
```
    $('#ff').form({
        success:function(data){
            $.messager.alert('Info', data, 'info');
        }
    });
```

服务器端代码

form1_proc.php

```
    $name = $_POST['name'];
    $email = $_POST['email'];
    $phone = $_POST['phone'];

    echo "Your Name: $name <br/> Your Email: $email <br/> Your Phor
```



下载 jQuery EasyUI 实例

jeasyui-form-form1.zip

jQuery EasyUI 表单 - 表单验证

本教程将向您展示如何验证一个表单。easyui 框架提供一个 validatebox 插件来验证一个表单。在本教程中，我们将创建一个联系表单，并应用 validatebox 插件来验证表单。然后您可以根据自己的需求来调整这个表单。

创建表单（form）

让我们创建一个简单的联系表单，带有 name、email、subject 和 message 字段：

```
<div style="padding:3px 2px;border-bottom:1px solid #ccc">Form
<form id="ff" method="post">
    <div>
        <label for="name">Name:</label>
        <input class="easyui-validatebox" type="text" name="nar
    </div>
    <div>
        <label for="email">Email:</label>
        <input class="easyui-validatebox" type="text" name="ema
    </div>
    <div>
        <label for="subject">Subject:</label>
        <input class="easyui-validatebox" type="text" name="sul
    </div>
    <div>
        <label for="message">Message:</label>
        &lt;textarea name="message" style="height:60px;"&gt;&lt
    </div>
    <div>
        <input type="submit" value="Submit">
    </div>
</form>
```

我们添加一个样式名为 easyui-validatebox 到 input 标记，所以 input 标记将根据 validType 属性应用验证。

当表单无效时阻止表单提交

当用户点击表单的 submit 按钮时，如果表单是无效的，我们应该阻止表单提交。

```
$('#ff').form({
    url:'form3_proc.php',
    onSubmit:function(){
        return $(this).form('validate');
    },
    success:function(data){
        $.messager.alert('Info', data, 'info');
    }
});
```

如果表单是无效的，将显示一个提示信息。

下载 jQuery EasyUI 实例

jeasyui-form-form3.zip

jQuery EasyUI 表单 - 创建树形下拉框

树形下拉框（ComboTree）是一个带有下列树形结构（Tree）的下拉框（ComboBox）。它可以作为一个表单字段进行使用，可以提交给远程服务器。

在本教程中，我们将要创建一个注册表单，带有 name、address、city 字段。city 字段是一个树形下拉框（ComboTree）字段，在里面用户可以下拉树面板（tree panel），并选择一个特定的城市。

创建表单（Form）

```
    <div id="dlg" class="easyui-dialog" style="width:500px;height:2
            title="Register" buttons="#dlg-buttons">
        <h2>Account Information</h2>
        <form id="ff" method="post">
            <table>
                <tr>
                    <td>Name:</td>
                    <td><input type="text" name="name" style="widtl
                </tr>
                <tr>
                    <td>Address:</td>
                    <td><input type="text" name="address" style="wi
                </tr>
                <tr>
                    <td>City:</td>
                    <td><select class="easyui-combotree" url="data/
                </tr>
            </table>
        </form>
    </div>
    <div id="dlg-buttons">
        <a href="#" class="easyui-linkbutton" iconCls="icon-ok" onc
        <a href="#" class="easyui-linkbutton" iconCls="icon-cancel'
    </div>
```

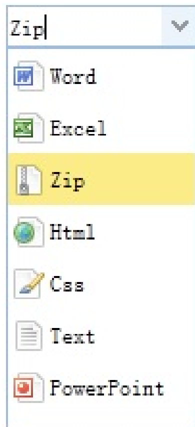
从上面的代码可以看到，我们为一个名为 'city' 的树形下拉框（ComboTree）字段设置了一个 url 属性，这个字段可以从远程服务器检索树形结构（Tree）数据。请注意，这个字段有一个样式名字叫 'easyui-combotree'，所以我们不需要写任何的 js 代码，树形下拉框（ComboTree）字段将自动渲染。

下载 jQuery EasyUI 实例

jeasyui-form-form2.zip

jQuery EasyUI 表单 - 格式化下拉框

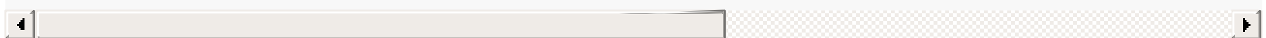
本教程向您展示如何创建一个简单的下拉框（Combobox），让它在下拉框中显示图片项。您可以在下拉框（combobox）上使用 formatter 函数来告诉它如何格式化每一个条目。



创建图像下拉框（Combobox）

```
<input id="cc" style="width:100px"
        url="data/combobox_data.json"
        valueField="id" textField="text">
</input>
```

```
$('#cc').combobox({
    formatter:function(row){
        var imageFile = 'images/' + row.icon;
        return '<img class="item-img" src="'+imageFile+'"/><spa
    }
});
```



下载 jQuery EasyUI 实例

jeasyui-form-form4.zip

jQuery EasyUI 表单 - 过滤下拉数据网格

下拉数据网格（Combogrid）组件和下拉框（Combobox）组件的共同点是，除了都具有下拉面板以外，它们都是基于数据网格（Datagrid）的。 下拉数据网格（Combogrid）组件可以过滤、分页，并具有其他一些数据网格（Datagrid）的功能。 本教程向您展示如何在一个下拉数据网格（Combogrid）组件中过滤数据记录。

K9

Item ID	Product ID	List Price	Unit Cost	Attribute	Stauts
EST-10	K9-DL-01	18.50	12.00	Spotted Adult Female	P
EST-22	K9-RT-02	135.50	100.00	Adult Male	P
EST-23	K9-RT-02	145.49	100.00	Adult Female	P
EST-24	K9-RT-02	255.50	92.00	Adult Male	P
EST-25	K9-RT-02	325.29	90.00	Adult Female	P
EST-26	K9-CW-01	125.50	92.00	Adult Male	P
EST-27	K9-CW-01	155.29	90.00	Adult Female	P
EST-28	K9-RT-01	155.29	90.00	Adult Female	P

创建下拉数据网格（Combogrid）

```
<input id="cg" style="width:150px">
```

```
$('#cg').combogrid({
    panelWidth:500,
    url: 'form5_getdata.php',
    idField:'itemid',
    textField:'productid',
    mode:'remote',
    fitColumns:true,
    columns:[[
        {field:'itemid',title:'Item ID',width:60},
        {field:'productid',title:'Product ID',align:'right',width:8
        {field:'listprice',title:'List Price',align:'right',width:6
        {field:'unitcost',title:'Unit Cost',align:'right',width:60
        {field:'attr1',title:'Attribute',width:150},
        {field:'status',title:'Stauts',align:'center',width:60}
    ]]
});
```

下拉数据网格（Combogrid）组件应该定义 'idField' 和 'textField' 属性。 'idField' 属性存储组件值，'textField' 属性在 input 文本框中显示文本消息。 下拉数据网格（Combogrid）组件可以以 'local' 或 'remote' 模式过滤记录。 在远程（remote）模式中，当用户输入字符到 input 文本框中，下拉数据网格（Combogrid）将发送 'q' 参数到远程服务器。

服务器端代码

form5_getdata.php

```
$q = isset($_POST['q']) ? strval($_POST['q']) : '';

include 'conn.php';

$rs = mysql_query("select * from item where itemid like '%$q%' or p
$rows = array();
while($row = mysql_fetch_assoc($rs)){
    $rows[] = $row;
}
echo json_encode($rows);
```

下载 jQuery EasyUI 实例

jeasyui-form-form5.zip

jQuery EasyUI 插件

jQuery EasyUI 提供了用于创建跨浏览器网页的完整的组件集合，包括功能强大的 datagrid（数据网格）、treegrid（树形表格）、 panel（面板）、combo（下拉组合）等等。 用户可以组合使用这些组件，也可以单独使用其中一个。

插件列表

Base（基础）

- Parser 解析器
- Easyloader 加载器
- Draggable 可拖动
- Droppable 可放置
- Resizable 可调整尺寸
- Pagination 分页
- Searchbox 搜索框
- Progressbar 进度条
- Tooltip 提示框

Layout（布局）

- Panel 面板
- Tabs 标签页/选项卡
- Accordion 折叠面板
- Layout 布局

Menu（菜单）与 Button（按钮）

- Menu 菜单
- Linkbutton 链接按钮
- Menubutton 菜单按钮
- Splitbutton 分割按钮

Form（表单）

- Form 表单
- Validatebox 验证框
- Combo 组合
- Combobox 组合框
- Combotree 组合树
- Combogrid 组合网格
- Numberbox 数字框

- Datebox 日期框
- Datetimebox 日期时间框
- Calendar 日历
- Spinner 微调器
- Numberspinner 数值微调器
- Timespinner 时间微调器
- Slider 滑块

Window（窗口）

- Window 窗口
- Dialog 对话框
- Messager 消息框

DataGrid（数据网格）与 Tree（树）

- Datagrid 数据网格
- Propertygrid 属性网格
- Tree 树
- Treegrid 树形网格

插件

easyui 的每个组件都有属性、方法和事件。用户可以很容易地对这些组件进行扩展。

属性

属性是定义在 jQuery.fn.{plugin}.defaults。比如，dialog 的属性是定义在 jQuery.fn.dialog.defaults。

事件

事件（回调函数）也是定义在 jQuery.fn.{plugin}.defaults。

方法

调用方法的语法：$('selector').plugin('method', parameter);

其中：

- selector 是 jquery 对象选择器。
- plugin 是插件名称。
- method 是与插件相匹配的已存在方法。
- parameter 是参数对象，可以是对象、字符串...

方法是定义在 jQuery.fn.{plugin}.methods。每个方法有两个参数：jq 和 param。第一个参数 'jq' 是必需的，引用 jQuery 对象。第二个参数 'param' 引用方法传递的实际参数。比如，要扩展 dialog 组件的方法名为 'mymove' 的方法，代码如下：

```
$.extend($.fn.dialog.methods, {
    mymove: function(jq, newposition){
        return jq.each(function(){
            $(this).dialog('move', newposition);
        });
    }
});
```

现在您可以调用 'mymove' 方法来移动对话框（dialog）到指定的位置：

```
$('#dd').dialog('mymove', {
    left: 200,
    top: 100
});
```

开始使用 jQuery EasyUI

下载库，并在您的页面中引用 EasyUI CSS 和 JavaScript 文件。

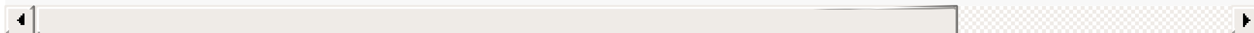
```
&lt;link rel="stylesheet" type="text/css" href="easyui/themes/defau
&lt;link rel="stylesheet" type="text/css" href="easyui/themes/icon.
&lt;script type="text/javascript" src="easyui/jquery-1.7.2.min.js"&
&lt;script type="text/javascript" src="easyui/jquery.easyui.min.js"
```

一旦您引用了 EasyUI 必要的文件，您就可以通过标记或者使用 JavaScript 来定义一个 EasyUI 组件。比如，要顶一个带有可折叠功能的面板，您需要编写如下 HTML 代码：

```
&lt;div id="p" class="easyui-panel" style="width:500px;height:200px
    title="My Panel" iconCls="icon-save" collapsible="true"&gt;
    The panel content
&lt;/div&gt;
```

当通过标记创建组件，'data-options' 属性被用来支持自版本 1.3 以来 HTML5 兼容的属性名称。所以您可以如下重写上面的代码：

```
&lt;div id="p" class="easyui-panel" style="width:500px;height:200px
    title="My Panel" data-options="iconCls:'icon-save',collapsible
    The panel content
&lt;/div&gt;
```

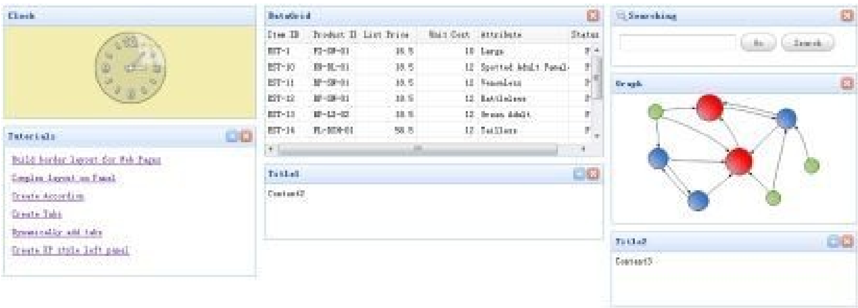


下面的代码演示了如何创建一个绑定 'onSelect' 事件的组合框。

```
&lt;input class="easyui-combobox" name="language"
    data-options="
    url:'combobox_data.json',
    valueField:'id',
    textField:'text',
    panelHeight:'auto',
    onSelect:function(record){
    alert(record.text)
    }"&gt;
```

jQuery EasyUI 扩展

Portal（制作图表、列表、球形图等）



数据网格视图（DataGrid View）

DataGrid - DetailView

	Item ID	Product ID	List Price	Unit Cost	Attribute	Status
+	EST-1	FI-SW-01	16.5	10	Large	P
+	EST-2	K9-DL-01	18.5	12	Spotted Adult Female	P
+	EST-3	RP-SN-01	18.5	12	Venomless	P
−	EST-4	RP-SN-01	18.5	12	Rattleless	P
	Attribute: Rattleless Status: P					
+	EST-5	RP-LI-02	18.5	12	Green Adult	P
+	EST-6	FL-DSH-01	58.5	12	Tailless	P

DataGrid - GroupView

	Product ID	List Price	Unit Cost	Attribute	Status
− K9-DL-01 - 1 Item(s)					
2	K9-DL-01	18.5	12	Spotted Adult Female	P
− RP-SN-01 - 2 Item(s)					
3	RP-SN-01	18.5	12	Venomless	P
4	RP-SN-01	18.5	12	Rattleless	P
+ RP-LI-02 - 1 Item(s)					
− FL-DSH-01 - 2 Item(s)					
6	FL-DSH-01	58.5	12	Tailless	P

可编辑的数据网格（Editable DataGrid）

Editable DataGrid

Item ID	Product ID	List Price	Unit Cost	Attribute
EST-1	FI-SW-01	16.5	10	Large
EST-2	K9-DL-01	18.5	12	Spotted Adult Female
EST-3	RP-SN-01	18.5	12	Venomless
EST-5	RP-LI-02	18.5	12	Green Adult
EST-6	FL-DSH-01	58.5	12	Tailless
EST-7	FL-DSH-01	23.5	12	With tail
EST-8	FL-DLH-02	93.5	12	Adult Female

可编辑的树（Editable Tree）

- Foods
  - Fruits
  - Vegetables
    - tomato
    - carrot
    - cabbage
    - potato
    - lettuce

数据网格行过滤（DataGrid Filter Row）

DataGrid

Item ID	Product	List Price	Unit Cost	Attribute	Status
		132.0	12.0		All
EST-1	FI-SW-01	3			P
EST-10	K9-DL-01	1		d Adult Female	P
EST-11	RP-SN-01	1		less	P
EST-12	RP-SN-01	2		ss	P
EST-13	RP-LI-02	3		Adult	P
EST-15	FL-DSH-01	83.5	12	With tail	P
EST-16	FL-DLH-02	23.5	12	Adult Female	P

- No Filter
- Equal
- Not Equal
- Less
- Greater

数据网格行拖放（Drag and Drop Rows in DataGrid）

DataGrid

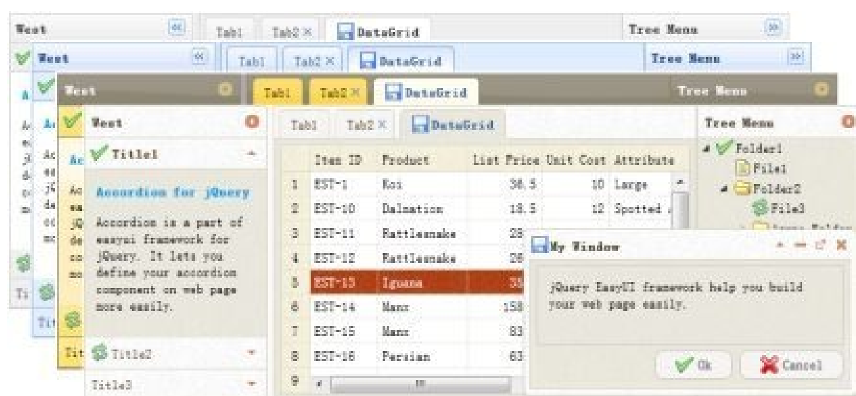
Item ID	Product	List Price	Unit Cost	Attribute	Status
EST-1	FI-SW-01	36.5	10	Large	P
EST-10	K9-DL-01	18.5	12	Spotted Adult Female	P
EST-11	RP-SN-01	38.5	12	Venomless	P
EST-12	RP-SN-01	26.5	12	Rattleless	P
EST-13					
EST-11	RP-SN-01	38.5	12	Venomless	P
EST-14	FL-DSH-01	158.5	12	Tailless	P
EST-15	FL-DSH-01	83.5	12	With tail	P
EST-16	FL-DLH-02	23.5	12	Adult Female	P

树形网格行拖放（Drag and Drop Rows in TreeGrid）

Folder Browser

	Name	Size	Modified Date
1	C		02/19/2010
2	Program Files	120 MB	03/20/2010
3	Java		01/13/2010
6	MySQL my-huge.ini		01/13/2010
7	my.ini	10 KB	02/26/2009
8	my-huge.ini	5 KB	02/26/2009
9	my-large.ini	5 KB	02/26/2009
10	eclipse.ini	1 KB	04/20/2010

主题（Themes）



DWR 加载器（DWR Loader）



RTL 支持（RTL support）

West | Title1 | Title2 | content2 | Title3

Main Title

DataGrid | About

Status	Attribute	Unit Cost	List Price	Product ID	Item ID
P	Large	10	36.5	FI-SW-01	EST-1
P	otted Adult Female	12	18.5	K9-DL-01	EST-10
P	Venomless	12	38.5	RP-SN-01	EST-11
P	Rattleless	12	26.5	RP-SN-01	EST-12
P	Green Adult	12	35.5	RP-LI-02	EST-13
P	Tailless	12	158.5	FL-DSH-01	EST-14

East

My Documents
Photos
Program Files
Intel
Java
Microsoft Office
Games
index.html
about.html
welcome.html

Ribbon

Home | Insert

Paste | Cut | Copy | Format | Clipboard

Arial | 12 | B | I | U | S | Font

Paragraph

Find | Find | Go to... | Editing

免责声明

W3School提供的内容仅用于培训。我们不保证内容的正确性。通过使用本站内容随之而来的风险与本站无关。W3School简体中文版的所有内容仅供测试，对任何法律问题及风险不承担任何责任。